



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

TELEKOMUNIKACIJŲ INŽINERIJOS KATEDRA

Arūnas STATKUS

**TCP PRANEŠIMŲ PATVIRTINIMO MECHANIZMO ĮTAKOS MARŠRUTO
PARINKTUVO NAŠUMUI TYRIMAS**

**ANALYSIS OF THE TCP MESSAGE ACKNOWLEDGMENT TECHNIQUE
INFLUENCE TO ROUTER PERFORMANCE**

Magistro baigiamasis darbas

Elektronikos inžinerijos studijų kryptis

Telekomunikacijų inžinerijos studijų programa, valstybinis kodas 62401T210

Telekomunikacijų technologijų specializacija

Vilnius, 2010

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
Telekomunikacijų inžinerijos katedra

TVIRTINU
Katedros vedėjas

(parašas)
prof. habil. dr. A. Kajackas
2010 m. _____ mėn. ____ d.

Arūnas STATKUS

**TCP PRANEŠIMŲ PATVIRTINIMO MECHANIZMO ĮTAKOS MARŠRUTO
PARINKTUVO NAŠUMUI TYRIMAS**
**ANALYSIS OF THE TCP MESSAGE ACKNOWLEDGMENT TECHNIQUE
INFLUENCE TO ROUTER PERFORMANCE**

Magistro baigiamasis darbas

Elektronikos inžinerijos studijų kryptis
Telekomunikacijų inžinerijos studijų programa, valstybinis kodas 62401T210
Telekomunikacijų technologijų specializacija

Vadovas	dr. Lukas Pavilanskas (mokslinis vardas ir laipsnis, vardas, pavardė)	_____	_____
		(parašas)	(data)
Konsultantas	doc. dr. Zofija Babickienė (mokslinis vardas ir laipsnis, vardas, pavardė)	_____	_____
		(parašas)	(data)

Vilniaus Gedimino technikos universitetas
Elektronikos fakultetas
Telekomunikacijų inžinerijos katedra

ISBN ISSN
Egz. sk.
Data-.....-.....

Antrosios pakopos studijų **Telekomunikacijų inžinerijos** programos baigiamasis darbas
Pavadinimas **TCP pranešimų patvirtinimo mechanizmo įtakos maršruto parinktuvo
našumui tyrimas**

Autorius **Arūnas Statkus**
Vadovas **dr. Lukas Pavilanskas**

Kalba: lietuvių

Anotacija

Baigiamajame magistro darbe išnagrinėtas TCP protokolo veikimas, aptarti pagrindiniai mechanizmai ir algoritmai, naudojami norint užtikrinti sėkmingą duomenų perdavimą. Yra aptartas ir išanalizuotas TCP ACK paketų informacijos perteklius ir jo priežastys. Atlikta analizė kaip šis perteklius yra mažinamas ir pasiūlytas metodas kaip sumažinti šį perteklių, t.y. naudojant nustatytų sąlygų ACK filtrą. Darbe atlikti praktiniai matavimai su nauju ACK filtravimo mechanizmu, parodyti šio metodo privalumai ir trūkumai.

Prie esamų ACK filtravimo metodų pasiūlytos naujo šio mechanizmo pritaikymo galimybės, kai ACK filtravimas yra naudojamas simetrinėse didelės spartos perdavimo linijose, norint sumažinti tarpinių ir galinių tinklo įrenginių (maršrutizatorių) apkrovą. Darbe atlikti praktiniai matavimai su skirtinga įranga, parodanti naujojo metodo teikiamus privalumus ir galimybes. Darbą sudaro 8 dalys: įvadas, TCP protokolo apžvalga, TCP ACK mechanizmas, ACK filtravimo įtaka TCP sesijai, ACK filtravimo įtaka maršruto parinktuvo apkrovai, išvados, literatūros sąrašas.

Darbo apimtis – 59 p. teksto be priedų, 30 iliustr., 8 lent., 29 bibliografiniai šaltiniai.
Atskirai pridedami darbo priedai.

Prasminiai žodžiai

Ethernet, TCP/IP, CUBIC, Reno, Vegas, HTCP, ACK filtravimas, Linux, Cisco, paketų praradimai, duomenų perdavimo sparta, maršruto parinktuvas, apkrovos mažinimas.

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Telecommunication Engineering

ISBN ISSN
Copies No.
Date-.....-.....

Master Degree Studies **Telecommunications Engineering** study programme Master's Thesis

Title **Analysis of the TCP Message Acknowledgment Technique Influence to Router Performance**

Author **Arūnas Statkus**

Academic supervisor **dr. Lukas Pavilanskas**

Thesis language:
Lithuanian

Annotation

Final master thesis presents the TCP protocol performance and reveals the basic mechanisms and algorithms which are used to ensure a successful data transfer. TCP ACK technological expenditures and it's causes broadly discussed and analyzed. Analysis has been performed with the purpose to find possibilities in minimizing above mentioned technological expenditures and to propose method to reduce this protocol overhead by using ACK filtering. Practical measurements with a new ACK filtering mechanism have been applied in the study in order to demonstrate the advantages and disadvantages of the method.

Adjustment of new mechanism possibilities has been introduced among the current ACK filtering methods especially when the ACK filtering is used in symmetrical, high-speed transmission lines, hoping to reduce the load of intermediate and terminal network devices (routers). Practical measurements with different equipment has been accomplished and presented in the final study with an intention to show the advantages and opportunities of the new method.

The study consists of 8 parts: introduction, overview of TCP, TCP ACK mechanism, TCP ACK filtering effect session, ACK filtering influence on router load, the conclusions and references.

Thesis consists of 59 p. of text without extras, 30 pictures, 8 tables, 29 bibliographical entries.

Appendixes included.

Keywords

Ethernet, TCP/IP, Reno, Vegas, HTCP, ACK filtering, Linux, Cisco, packet loss, data transmission speed, router, load reductions

TURINYS

Žymenys ir santrumpos	7
1. ĮVADAS	8
1.1. Darbo aktualumas ir tikslas	8
1.2. Darbo uždaviniai.....	8
1.3. Darbo naujumas ir praktinė nauda.....	9
2. TRANSPORTINIO LYGMENS PROTOKOLO APŽVALGA	10
2.1. TCP protokolas pagal OSI kompiuterinių tinklų modelį.....	10
2.2. TCP protokolo struktūra	10
2.3. Šaltinio ir paskirties prievadai	11
2.4. Paketo eiliškumo ir patvirtinimo laukai.....	12
2.5. TCP antraštės žymės ir jų funkcijos	13
Pirmumo rodyklės požymis (URG).....	13
Patvirtinimo požymis (ACK).....	14
Skubaus siuntimo požymis (PSH)	14
Nutraukimo požymis (RST)	14
Sinchronizavimo požymis (SYN).....	14
Duomenų siuntimo pabaigos požymis (FIN).....	15
Duomenų lango sumažinimo ir tinklo perkrovos požymiai (CWR, ECN).....	15
2.6. TCP srauto kontrolės mechanizmai	16
2.7. Duomenų patikros suma	17
2.8. Pirmumo rodyklė	18
2.9. TCP papildomi parametrai.....	18
3. TCP DUOMENŲ PERDAVIMO KONTROLĖ	20
3.1. TCP duomenų gavimo ir siuntimo langai (CWIN, WIN)	20
3.2. Lėto starto fazė	21
3.3. Persipildymo vengimo fazė	21
3.4. Tinklo vėlinimo įtaka TCP veikimui	22
3.5. Pakartotinio persiuntimo laikas (RTO).....	26
4. TCP LANGO PERSIPILDYMO VENGIMO ALGORITMAI	27
4.1. TCP CUBIC algoritmas.....	27
BIC lango didėjimo funkcija	27
CUBIC lango didėjimo funkcija.....	28
5. ACK MECHANIZMAS	30
5.1. TCP ACK patvirtinimo mechanizmas	30
5.2. ACK pertekliaus mažinimas	32
5.3. ACK kompresija	32
5.4. Pavėlintas ACK (Delayed ACK).....	33
5.5. ACK filtravimas	33
6. ACK FILTRAVIMO MECHANIZMAI	35
6.1. Nustatytų sąlygų ACK filtravimas	35
Testavimo aplinka.....	36
6.2. ACK filtravimo įtaka TCP sesijos stabilumui	37
6.3. ACK filtravimas su keliomis TCP sesijomis.....	41
6.4. ACK filtravimo įtaka vienai TCP sesijai.....	44
6.5. TCP ACK filtravimas nehomogeninėse aplinkose.....	47
6.6. ACK filtravimo įtaka trumpų sesijų spartai.....	48
6.7. ACK filtravimas su skirtingais persipildymo vengimo algoritmais	50

7. ACK FILTRAVIMO ĮTAKA maršruto parinktuvo APKROVAI	52
7.1. ACK filtravimo įtaka maršruto parinktuvo apkrovai	52
8. Apibendrinimas. Išvados	57
Literatūra	59
PRIEDAI.....	61
A priedas. Pranešimo Evaluation of TCP acknowledgment mechanism influence on router performance. TELECOMMUNICATIONS ENGINEERING – T 180.....	61

Žymenys ir santrumpos

BIC (angl. <i>Binary Increase</i>)	TCP lango persipildymo vengimo funkcija
CBS (angl. <i>Committed Burst Size</i>)	leidžiamas duomenų pliūpsnis
CIR (angl. <i>Committed Information Rate</i>)	leidžiama informacijos perdavimo sparta
CUBIC (angl. <i>Cubic</i>)	TCP lango persipildymo vengimo funkcija
CWIN (angl. <i>Congestion Window</i>)	nurodo maksimalų duomenų kiekį, kurį gali išsiųsti TCP duomenų siuntėjas negavęs patvirtinimo iš TCP duomenų gavėjo
FIFO (angl. <i>First In First Out</i>)	duomenų buferio apdorojimo algoritmas
FTP (angl. <i>File transfare Protocol</i>)	reikmenų apsikeitimo protokolas
IP (angl. <i>Internet Protocol</i>)	IP protokolas
IPsec (angl. <i>IP security</i>)	IP duomenų apsaugos protokolas
MMS (angl. <i>Maximu Segment Size</i>)	maksimalus segmento dydis
OSI (angl. <i>Open Systems Interconnection Reference Model</i>)	atvirųjų sistemų tarpusavio ryšio modelis
Policer (angl. <i>Traffic Policer</i>)	nustatytą sąlygų duomenų srauto ribojimas
RTO (angl. <i>retransmission timeout</i>)	pakartotinio persiuntimo laikas
RTT (angl. <i>Round TripTime</i>)	duomenų siuntimo vėlinimas į abi puses
S/MIME (angl. <i>Secure/Multipurpose Internet Mail Extensions</i>)	šifravimo ir autentifikavimosi algoritmas
Shaper (angl. <i>Traffic Shaper – Shaper</i>)	spartos ribojimo mechanizmas
SlowStart (angl. <i>Slow Start – SlowStart</i>)	TCP lėto starto fazė
SSL (angl. <i>Secure Sockets Layer</i>)	kriptografinis protokolas, skirtas informacijos, sklindančios internete apsaugojimui šifruojant
SSTHRESH (angl. <i>Slow Start Threshold</i>)	persipildymo lango slenkstinė vertė
TCP (angl. <i>Transport Control Protocol</i>)	patikimas duomenų perdavimo protokolas
TCP/IP mode	tarpusavio protokolo ir sistemų sąryšio modelis
UDP (angl. <i>User Datagram Protocol</i>)	nepatvirtinamas duomenų perdavimo protokolas
WIN (angl. <i>Window Size</i>)	nurodo maksimalų duomenų kiekį, kurį gali priimti TCP duomenų gavėjas

1. ĮVADAS

1.1. Darbo aktualumas ir tikslas

Dėl sparčiai tobulėjančių technologijų duomenų perdavimo sparta per pastarąjį dešimtmetį padidėjo daugiau nei šimtą kartų. Norint efektyviai išnaudoti duomenų srautus ir patikimai perduoti informaciją, yra naudojamas TCP protokolas.

TCP protokolas buvo sukurtas daugiau nei prieš 30 metų norint užtikrinti patikimą duomenų perdavimą ir tolygų tinklo išteklių padalinimą tarp vartotojų. Didėjant perdavimo spartoms ir atsirandant skirtingoms duomenų perdavimo technologijoms vis daugiau dėmesio skiriama TCP protokolo efektyvumo didinimui. Šioje srityje daug dirba Van Jacobson [1–3], kuris 80-ųjų pabaigoje pasiūlė *Tahoe/Reno* TCP algoritmą. Jo pasiūlytas TCP protokolo persipildymo vengimo algoritmas sukurtas laikantis taškas – taškas algoritmo principo ir neleido internetui „užsikimšti“. Kiti TCP protokolo persipildymo vengimo algoritmai: *Reno*, *New Reno*, *Vegas*, *Westwood+*, *BIC* sprendžia analogiškus uždavinius [3–5].

Kuriant TCP protokolą buvo stengiamasi sukurti kuo universalesnį duomenų perdavimo protokolą, kuris galėtų veikti efektyviai skirtingose aplinkose ir situacijose. Tačiau norint iki galo išnaudoti TCP protokolo teikiamus privalumus, reikia kuo daugiau žinoti pradinės informacijos apie duomenų perdavimo tinklą, kuriame jis veiks. Netinkami ir blogai parinkti parametrai ar persipildymo vengimo algoritmai gali ne tik sumažinti perdavimo spartą, bet ir paveikti visą tinklą ir jo vartotojus. Norint kuo optimaliau išnaudoti tinklo išteklius ir sudaryti tolygų srauto padalinimą tarp atskirų vartotojų reikia tiksliai žinoti kiekvieno iš TCP persipildymo vengimo algoritmų veikimo ypatumus, jų pranašumus ir trūkumus.

1.2. Darbo uždaviniai

TCP protokolas pasižymi savo universalumu, todėl jis yra naudojamas tiek didelio vėlinimo palydovinėse sistemose, tiek ir didelės spartos optiniuose tinkluose. Kaina, kurią mes mokame už šį universalumą, yra didelis protokolo informacijos perteklius, kuris yra būtinas sudėtingose aplinkose, tačiau visai nebūtinai duomenų perdavimo tinkluose su nuliniu duomenų praradimu ir mažu vėlinimu.

Tiriamąjį darbo tikslas yra ištirti vieną iš galimų TCP pertekliaus mažinimo galimybių, t.y. nustatytų sąlygų ACK filtravimą, išnagrinėti kokią įtaką tai daro TCP duomenų perdavimo spartai, interaktyvumui ir kaip tai daro įtaka tarpinius tinklo įrenginius.

Darbo metu atlikti praktiniai tyrimai ir matavimai norint geriau suprasti ACK filtravimo mechanizmą, jo įtaką TCP sesijos stabilumui ir spartai. Praktiškai parodyta maksimali ACK filtravimo

vertė ir jos sąryšis su perdavimo sparta, pateiktos galimos problemos naudojant pasiūlytą ACK filtravimo mechanizmą.

1.3. Darbo naujumas ir praktinė nauda

Tiriamąo darbo metu pasiūlyta visiškai nauja ACK filtravimo panaudojimo sritis, kai šis mechanizmas yra naudojamas norint sumažinti tinklo įrenginių apkrovą.

Didžioji dalis tinklo įrenginių (ugniasienių, maršruto parinktųjų, serverių ir t.t.) apkrova yra tiesinė ir priklauso nuo gaunamų ir apdorojamų paketų skaičiaus. Norint sumažinti jų apkrovą ir padidinti TCP duomenų perdavimo spartą ACK filtravimas gali būti naudojamas.

2.TRANSPORTINIO LYGMENS PROTOKOLO APŽVALGA

Šiame skyriuje apžvelgtas TCP protokolas, jį sudarantys laukai, reikšmės bei jų paskirtis. Papildomai pateikti standartiniai TCP algoritmai, jų veikimo principai ir paskirtis. Didesnioji skyriaus dalis skirta TCP duomenų perdavimo ir patvirtinimo veikimo mechanizmams, kurie yra atsakingi už kokybišką duomenų perdavimą tarp nutolusių TCP sesijos narių.

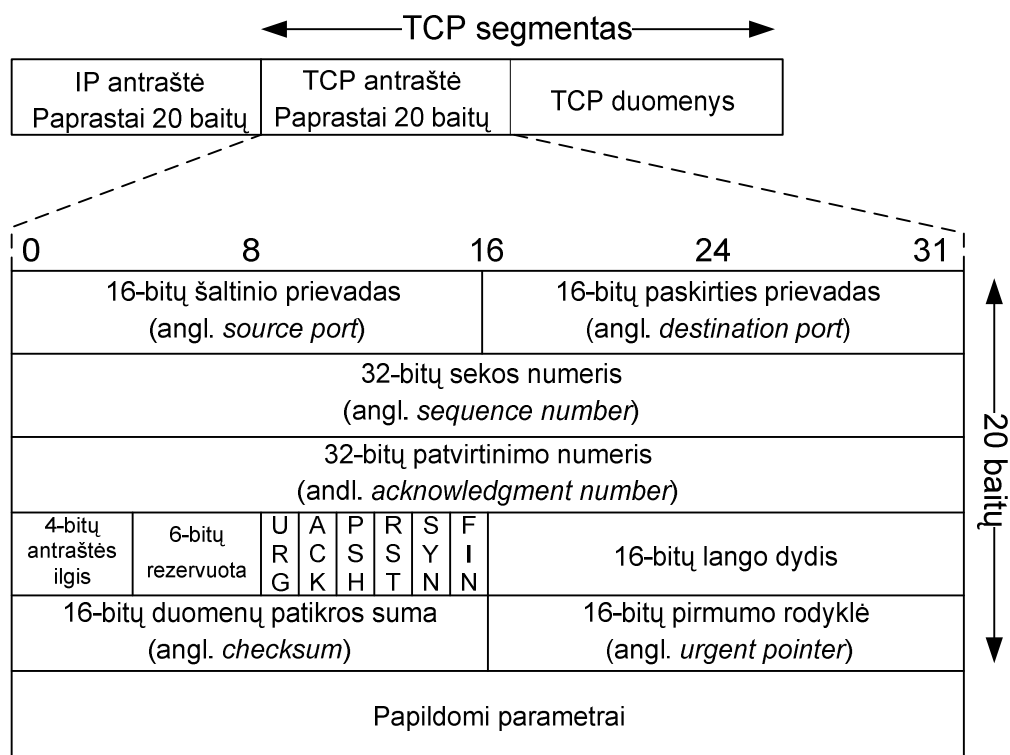
2.1. TCP protokolas pagal OSI kompiuterinių tinklų modelį

TCP protokolas atsiradęs daugiau nei prieš trisdešimt metų yra vienas iš populiariausių ir dažniausiai naudojamų duomenų perdavimo protokolų internete. Pagal atvirų sistemų modelį – OSI, TCP protokolas yra priskiriamas ketvirtajam transportavimo lygiui, kuris aprašo duomenų perdavimą tarp tinklo elementų – vartotojų. Tačiau kalbant apie TCP protokolą, reikėtų jį nagrinėti TCP/IP modelio pagrindu. Nors OSI modelis yra tarptautinis standartas, atsiradęs 1983 metais, tačiau jis yra retai naudojamas praktikoje.

2.2. TCP protokolo struktūra

Transportavimo kontrolės protokolo paskirtis yra užtikrinti patikimą ir kontroliuojamą duomenų perdavimą terpę tarp dviejų nutolusių įrenginių. Naudodamos šį protokolą, taikomosios programos sukuria logines jungtis, per kurias vyksta informacijos apsikeitimas. Priešingai nei UDP, šis protokolas užtikrina patikimą duomenų perdavimą tarp dviejų tinklo taškų. Tokiu būdu aukštesnio lygio protokolams bei programoms nereikia rūpintis dėl duomenų perdavimo ar galimų klaidų, kurios gali atsirasti perduodant duomenis per tinklą. Nors TCP protokolas užtikrina patikimą duomenų perdavimą per tinklą, tačiau jis neturi priemonių apsaugoti duomenis nuo jų peržiūrėjimo ar jų pakeitimo trečiųjų asmenų, tai yra tik transportavimo protokolas. Saugiam duomenų perdavimui su TCP protokolu yra naudojami papildomi protokolai, tokie kaip IPsec ar aukštesnio lygmens protokolai – SSL ar S/MIME.

Standartiškai TCP yra naudojamas su IP protokolu, kuris yra atsakingas už paketų adresavimą, fragmentavimą ir maršruto parinkimą. Tačiau IP protokolas negali užtikrinti, kad duomenys pasieks adresatą. Norėdami suprasti TCP veikimą, visų pirma reikia susipažinti su protokolo antrašte ir jos laukų paskirtimi. TCP protokolo antraštė sudėtinga ir gali skirtis skirtinguose sistemose ar veikimo fazėse, tačiau tipinė TCP antraštė pateikta 1 pav.



1 pav. TCP antraštė

Nagrinėdami TCP protokolą jo antraštės suskirstysime į šešias logines dalis:

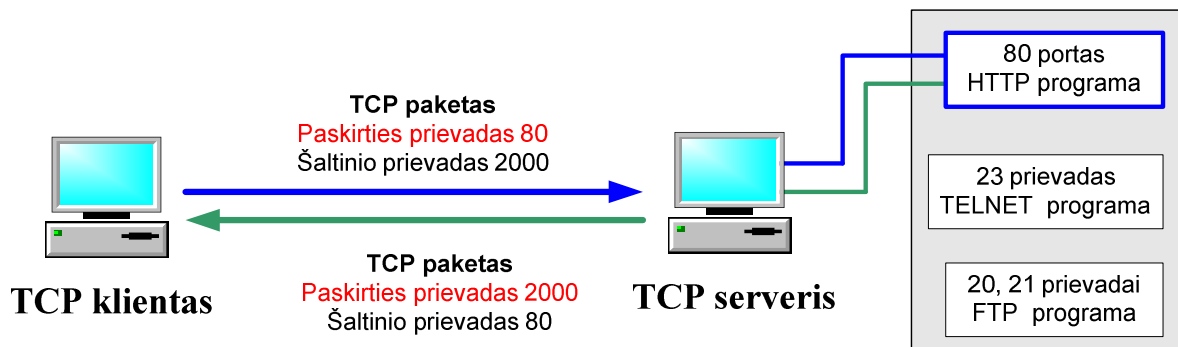
1. Šaltinio ir paskirties prievadai;
2. Paketo eiliškumo ir patvirtinimo laukai;
3. Antraštės ilgio laukas;
4. TCP požymių laukai;
5. TCP lango, duomenų patikros ir pirmumo rodyklės laukai;
6. TCP papildomi parametrai.

2.3. Šaltinio ir paskirties prievadai

Kuriant TCP protokolą, daug dėmesio buvo skiriama jo būsimai raidai ir universalumui. Vienas iš to pavyzdžių yra šaltinio ir paskirties laukai – prievadai. Kiekvienas iš jų yra 16 bitų ilgio ir tai leidžia sukurti 65 536 skirtingus prievadus.

Viešasis IP adresas yra unikalus kiekvienam tinklo įrenginiui [4], tačiau jis nieko nepasako apie paslaugas, kurios veikia tame įrenginyje. Tam tikslui yra naudojami TCP ar UDP paskirties ir šaltinio prievadai. Kai mes norime prisijungti prie HTTP puslapio, siunčiame TCP paketą su paskirties (angl. *destination*) prievadu 80. Serveris, gavęs paketą pasižiūri į paskirties prievadą ir perduoda atitinkamai programai aptarnauti, mūsų atveju tai HTTP, kuris veikia 80 prievade.

Paskirties prievadas aprašo programą ar paslaugą, kuri veikia serveryje/kompiuteryje ir naudojama bendrauti su kitais tinklo elementais.



2 pav. TCP šaltinio ir paskirties prievadais

Kadangi paskirties prievadais aprašo norimą pasiekti paslaugą ar programą, tai šaltinio prievadas nurodo paslaugai (TCP serveriu), į kokią prievadą reikia siųsti TCP atsakymą.

Serveris norėdamas atsakyti į gautą TCP paketą turi sugeneruoti naują paketą, kuriame paskirties (80) ir šaltinio (2000) prievadais yra sukeisti vietomis (2 pav.).

2.4. Paketo eiliškumo ir patvirtinimo laukai

TCP yra transportavimo protokolas, todėl norint užtikrinti, kad kiekvienas išsiųstas duomenų bitas tikrai pasieks adresatą mes turime atlikti duomenų ir paketų numeravimą, sekimą. Tam tikslui naudojamas TCP „eiliškumo“ (angl. *sequence* – SEQn) laukas. Tai yra 32 bitų ilgio laukas, einantis po šaltinio ir paskirties laukų.

Jis nurodo ne tik paketo eiliškumo numerį, bet ir kiek duomenų yra gavęs siuntėjas bei informacijos kiekį, kurį perneša paketas. Priešingai nei daugelis galvoja, eiliškumo numeris didėja ne nuosekliai, bet priklausomai nuo perduotų duomenų kiekio, (išskyrus tik sesijos sudarymo ir nutraukimo fazes). Visa tai atliekama dėl žemiau išdėstytų priežasčių:

1. Atliekamas TCP paketų numeravimas;
2. Tiksliai žinoma kiek duomenų yra perduodama TCP paketu ir jai įvyksta paketo praradimas, tiksliai žinome, kurie duomenys dingę. Tai daroma siekiant taupyti duomenų siuntėjui atminties išteklius [4]. Nes priešingu atveju TCP paketus siunčiantysis įrenginys turėtų laikyti visus nepatvirtintus paketus atmintyje tol, kol nebus gautas patvirtinamasis paketas – TCP ACK;

3. Kiekvieną kartą sudarant naują TCP sesiją, paketo eiliškumo numeris yra suteikiamas atsitiktinai. Tai užtikrina, jog TCP sesija bus unikali bei sumažina tikimybę, kad TCP sesiją gali perimti trečiasis asmuo.

Tačiau klientui žinoti, kiek duomenų jis yra gavęs ir ar nėra prarastų paketų, nėra prasminga, jei apie tai neinformuojame paketų siuntėją. Šį darbą atlieka TCP patvirtinimo (angl. *acknowledgment* – ACKn) laukas. TCP duomenų siuntėjas, gavęs šią informaciją tiksliai žino kiek duomenų turi klientas ir ar nereikalingas papildomas duomenų persiuntimas. Į siunčiamo patvirtinimo ACKn lauką įrašomas prieš tai gauto paketo eiliškumo numeris – $SEQn + D$, kur D atitinka paskutinio gauto TCP paketo duomenų kiekį baitais. Gautoji eiliškumo reikšmė turi dvigubą prasmę (ACKn nurodo kiek tiksliai klientas yra gavęs duomenų ir koks „turėtų būti“ kitas serverio paketo eiliškumo numeris $SEQn$):

1. Nurodo duomenų kiekį baitais, kuriuos priėmė TCP klientas;
2. Nurodo, koks turi būti kito TCP paketo sekos numeris.

2.5. TCP antraštės žymės ir jų funkcijos

Norint išskirti TCP protokolo paketus iš bendro srauto ar suteikti jam papildomą prioritetą, naudojami TCP požymio laukai. Šie laukai nustato papildomą TCP paketo paskirtį ir reikšmę TCP sesijoje. Tai gali būti sesijos sudarymo ar nutraukimo parametrai. Mes galime pažymėti vienus duomenų paketus kaip svarbius arba nutraukti sesiją, jei kyla problemų.

TCP paketo požymio laukas yra 6 bitų ilgio, kiekvienas atskiras bitas atitinka tam tikrą reikšmę. Žemiau pateiktas jų vertimas ir trumpas jų paskirties aprašymas:

1. Pirmumo rodyklės požymis (angl. *urgent* – URG);
2. Sinchronizavimo, susirišimo požymis (angl. *synchronize* – SYN);
3. Patvirtinimo požymis (angl. *acknowledgment* – ACK);
4. TCP sesijos nutraukimo požymis (angl. *reset* – RST);
5. Duomenų siuntimo pabaigos požymis (angl. *finish* – FIN);
6. Prioriteto, svarbumo požymis (angl. *push* – PSH);
8. Duomenų siuntimo lango sumažinimo (angl. *Congestion Window Reduced* – CWR) ir tinklo perkrovos (angl. *Explicit Congestion Notification* – ECN) požymiai.

Pirmumo rodyklės požymis (URG)

Tai pirmasis TCP požymio lauko parametras nurodantis, kad gautas paketas yra svarbus/skubus ir jį reikia apdoroti kaip įmanoma greičiau, anksčiau už kitus. Toks parametras gali būti panaudotas norint staigiai nutraukti aktyvią TCP sesiją [4].

TCP klientas ar serveris, norintis nutraukti aktyvią TCP sesiją, išsiunčia paketą su FIN požymiu. Gavęs šį paketą serveris/klientas stato šį paketą į FIFO eilę, kur visi paketai apdorojami iš eilės.

Norint to išvengti ir kuo greičiau nutraukti TCP sesiją, galime pasinaudoti TCP URG požymiu. Serveris ar klientas, gavęs šį paketą, jį apdoroja pirmiausiai, apeinant FIFO duomenų eilę, taip sutaupydamas sisteminius ir laiko išteklius.

Patvirtinimo požymis (ACK)

Patvirtinimo požymiu siuntėjas informuojamas apie sėkmingai gautus duomenis. Vykstant TCP sesijai apsiikeičiama informacija – duomenimis, norint patvirtinti, kad siųstas paketas yra gautas, gavėjas išsiunčia TCP paketą su nustatytu ACK požymiu ir į patvirtinimo lauką įrašo gautų duomenų kiekį. TCP ACK požymis gali būti naudojamas patvirtinti keliems paketams iš eilės siunčiant paskutinio priimto paketo eilės numerį + D [4].

Skubaus siuntimo požymis (PSH)

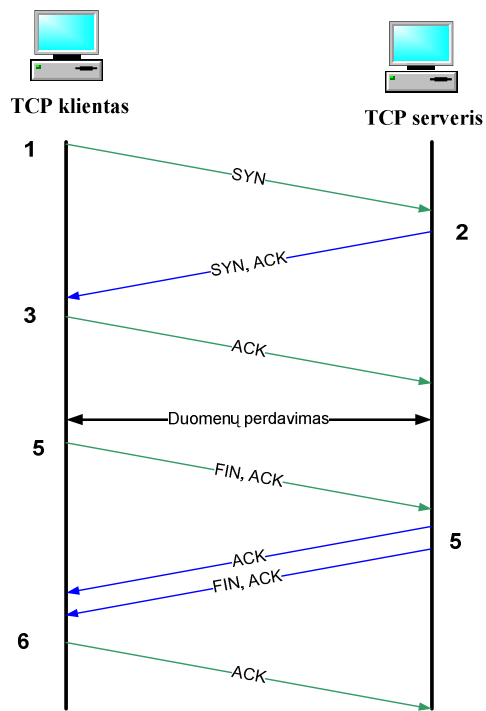
Šis požymis pagal savo paskirtį labai panašus į pirmumo rodyklės požymį, tačiau jis tik nurodo, kad duomenis reikėtų apdoroti kaip įmanoma greičiau. Kadangi daugelis šiuolaikinių OS tausoja sisteminius išteklius, tai duomenys į tinklą patenka ne iš karto. Norint išsiųsti TCP paketą, duomenys patenka į TCP laikiną atmintį, kurioje jie kaupiami. Sukaupus pakankamai duomenų ir užpildžius TCP segmentą paketas yra išsiunčiamas į tinklą. Tačiau daugeliui programų ar paslaugų (telnet, internetinių pokalbių, realaus laiko programa), veikiančių TCP pagrindu, reikalingas kuo greitesnis duomenų perdavimas. Norint išvengti nereikalingo užlaikymo TCP buferyje, yra naudojamas PSH požymis, kuris praneša, kad duomenys turi būti apdorojami apeinant laikiną [4] TCP talpinimo atmintį (buferį). Praktikoje dažniausiai paskutinis TCP paketas turi PSH požymį, norint kuo skubiau užbaigti TCP sesiją ir atlaisvinti sisteminius išteklius.

Nutraukimo požymis (RST)

Nutraukimo požymis yra naudojamas norint nutraukti ar informuoti kitą TCP sesijos narį apie sesijos nutraukimą. To priežastys gali būti įvairios – nuo sisteminių išteklių trūkumo iki programos sustabdymo ar sistemos perkrovimo. Tokiu būdu sesijos iniciatorius žino, kad sesija yra negalima ir nutraukia naujos TCP sesijos sudarymo procesą arba nustoja siųsti duomenis.

Sinchronizavimo požymis (SYN)

Šis požymis naudojamas tik naujos sesijos sudarymo metu, informuojant kitą TCP sesijos narį, kad norima sudaryti TCP ryšį (angl. *Three Way Handshake – ThreeWay Handshake*). Tai trijų dalių sesijos sudarymo mechanizmas, pavaizduotas 3 pav.



3 pav. TCP sesijos sudarymas ir nutraukimas

Toliau aptarsime šio mechanizmo veikimo principus:

1. Kompiuteris A, norintis „bendrauti“ su kompiuteriu B siunčia TCP paketą su SYN požymiu;
2. Kompiuteris B, atsako TCP paketu su SYN ir ACK požymiais, sakydamas, kad jis irgi nori „bendrauti“;
3. Kompiuteris A, gavęs šį paketą, išsiunčia TCP pranešimą su ACK požymiu, sakydamas kad paketas gautas, ir galima pradėti duomenų apsikeitimą.

Duomenų siuntimo pabaigos požymis (FIN)

Šis požymis yra naudojamas norint užbaigti aktyvią TCP sesiją, t.y., kuomet baigus duomenų apsikeitimą reikia sustabdyti TCP sesiją [4]. Pabaigos požymis turi būti aktyvuotas abiejuose tinklo įrenginiuose, nes TCP sesija yra dvikryptė ir duomenų siuntimas turi vykti abiem kryptimis. Pasiuntus nutraukimo signalą, antras TCP sesijos narys vis dar gali siųsti duomenis ir sesija bus aktyvi tol, kol jis neišsiųs FIN požymio. Nutraukimo požymio veikimas pateiktas 3 pav.

Duomenų lango sumažinimo ir tinklo perkrovos požymiai (CWR, ECN)

Nors šie požymiai priklauso 6 bitų rezervuotam TCP laukui, tačiau dauguma programų ir literatūros šaltinių juos priskiria prie TCP požymių lauko.

Duomenų siuntimo lango ir tinklo perkrovos požymiai buvo aprašyti RFC 3168 dokumente ir yra naudojami kartu. Jų paskirtis yra efektyvesnis duomenų kanalo išnaudojimas, reguliuojant duomenų išsiuntimo greitį. CWR ir ECE laukų veikimas pagrįstas pastoviu duomenų siuntėjo informavimu apie

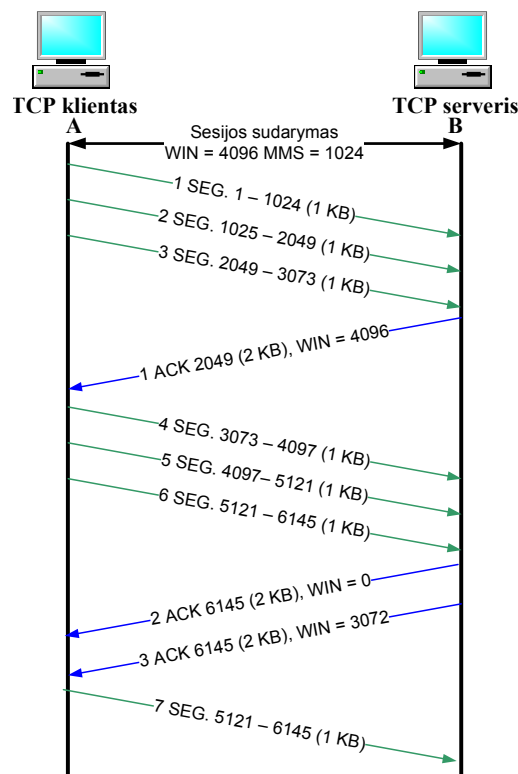
tinklo išteklius ir TCP duomenų gavėjo būklę. Tam kad būtų galima pasinaudoti šiais požymiais, reikalingas duomenų tinklo palaikymas, kuris galėtų informuoti apie tinklo perkrovas ECN požymiu.

Nors šis mechanizmas buvo dokumentuotas 2001 metais išleistame RFC 3168 dokumente, tačiau dėl sudėtingumo (visi tarpiniai tinklo įrenginiai turi palaikyti ECN) ir saugumo problemų yra retai taikomas praktikoje.

2.6. TCP srauto kontrolės mechanizmai

Kuriant TCP protokolą praeitame 70 dešimtmetyje daug dėmesio buvo skiriama ne tik efektyviam ir patikimam protokolo veikimui, bet ir TCP klientų ir serverio sisteminių išteklių valdymui. Norint kuo efektyviau išnaudoti ne tik duomenų perdavimo tinklą [5], bet ir galinius įrenginius – naudojamas TCP duomenų lango parametras (angl. *window* – WIN). Parametras yra 16 bitų ilgio. Juo duomenų gavėjas (B) informuoja duomenų siuntėją (A) apie tai, kiek maksimaliai duomenų (baitais) gali jis priimti vienu metu. Išsiuntus leidžiamą kiekį, TCP duomenų siuntėjas turi sustoti ir palaukti patvirtinimo – ACK paketo iš gavėjo. Nors maksimalus TCP WIN parametro dydis negali viršyti $2^{16} = 65\,536$ B vertės (šiuolaikinėse sistemose tai yra nepakankamas dydis), tačiau pasinaudojus TCP WIN lango daugikliu (angl. *Window scale*), jis gali būti padidintas iki 1 GB.

Pateiktame 4 pav. pavaizduotas duomenų apsikeitimas, kai TCP klientas (A) siunčia duomenis į serverį B.



4 pav. TCP sesija ir WIN parametras

TCP sesijos sudarymo metu klientas ir serveris nustato maksimalų duomenų segmentą (angl. *Maximum Segment Size* – MSS) bei duomenų gavimo langą (angl. *Window Size* – WIN). Pirmasis parametras reikalingas norint nustatyti IP protokolo maksimalų neskaidoma paketo dydį. Antrasis WIN parametras yra atsakingas už leidžiamą lango dydį, kuris nurodo kiek duomenų galima išsiųsti negavus patvirtinimo. Pastarąjį parametą nurodo gavėjas.

Panagrinėkime 4 pav. pateiktą TCP sesiją ir WIN parametro veikimą. TCP sesijos sujungimo metu buvo nustatytos $WIN = 4096$ B ir $MSS = 1024$ B reikšmės. Kai klientas išsiuntė 1, 2 ir 3 segmentus, serveris atsiunčia patvirtinimą (1 ACK), kad gavo 1 ir 2 segmentus. Tada klientas siunčia 4, 5 ir 6 segmentus, tačiau serveris dar nespėjo apdoroti 3 segmento, todėl jis atsiunčia patvirtinimo paketą 2 (angl. *Acknowledgment* – ACK), kuriuo informuoja klientą, kad jis gavo 4, 5, 6 segmentus, bet dėl nespėto apdoroti 3 segmento jis nustato lango dydį lygų nuliui ($WIN = 0$). Tokiu būdu klientas nustoja siųsti duomenis iki tol, kol serveris neapdoros visų gautų segmentų. Kai visi segmentai jau apdoroti, serveris išsiunčia pranešimą 3 ACK paketu, kad jis pasiruošęs priimti naujus segmentus ir nustatomas naujas lango dydis ($WIN = 3072$ B). Jis sumažina maksimalų galimą paketų skaičių tinkle bei apsaugo serverį nuo duomenų perkrovimo.

Naudojant TCP WIN parametą, galima reguliuoti išsiunčiamų duomenų kiekį – spartą. Šis TCP mechanizmas labai aktualus sistemose su ribotais atminties išteklius ar esant stipriai apkrovai. Tačiau asmeniniams kompiuteriams, turintiems pakankamai sisteminių ir atminties išteklių, šis parametras dažniausiai nustatomas maksimalus.

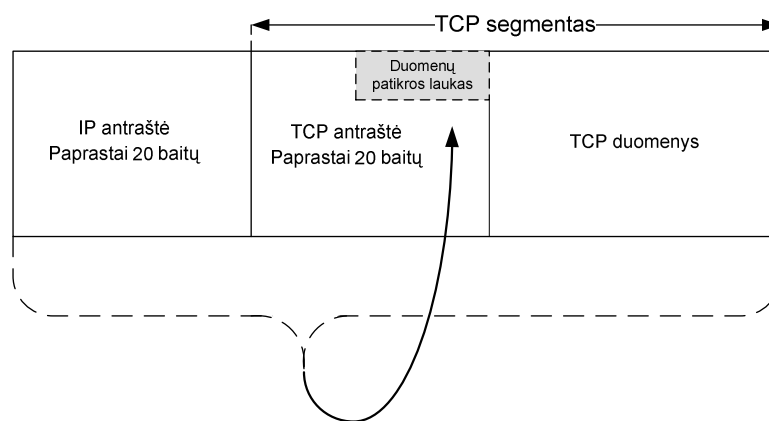
Nors WIN parametras nurodo maksimalų duomenų kiekį, kurį gali priimti TCP gavėjas, tačiau siunčiančioji pusė niekada neperduoda maksimalaus kiekio paketų iš karto. Tai daroma norint išvengti tinklo perpildymo, nes nei duomenų siuntėjas ar gavėjas nežino apie tinklo perdavimo spartą ir esamą apkrovą.

2.7. Duomenų patikros suma

Duomenų patikros laukas (angl. *checksum*) yra atsakingas už persiunčiamų duomenų integralumą, t.y. , kad duomenys pasiektų galutinį tikslą nepakitę ir nesugadinti. Sudarant duomenų patikros sumą be TCP paketo duomenų yra įtraukiamos ir kai kurios skiltys iš IP protokolo antraštės [4]. Tai yra daroma ne tik norint užtikrinti patikimą duomenų vientisumą, bet ir užtikrinti, kad duomenys tikrai atkeliavo iš norimo duomenų siuntėjo.

TCP duomenų patikros suma skaičiuojama prie TCP segmento pridėjus papildomus IP antraštės laukus:

1. IP paketo paskirties adresas (angl. *Source address*);
2. IP paketo paskirties adresas (angl. *Destination address*);
3. Rezervuotas laukas (dažniausiai būna visi nuliai) (angl. *Reserver*);
4. IP paketo protokolo laukas (angl. *Protocol*);
5. TCP segmento dydis, sudarytas iš TCP antraštės ir duomenų (angl. *TCP length*).



5 pav. TCP duomenų patikros skaičiavimas

Gautoji 16 bitų vertė įrašoma į TCP duomenų patikros lauką (5 pav.).

2.8. Pirmumo rodyklė

Pirmumo rodyklė (angl. *Uger pointer*) yra 16 bitų ilgio laukas, kuris yra naudojamas kartu su TCP URG požymiu. Kaip žinome, URG nurodo, kad paketas yra skubus [4]. Naudojant pirmumo rodyklės lauką papildomai nurodoma, kurie duomenys (baitai) yra svarbūs.

2.9. TCP papildomi parametrai

Nuolat tobulėjant duomenų perdavimo tinklams ir technologijoms, vis didesni reikalavimai yra keliami esamiems protokolams ir jų suderinamumui. Tam tikslui TCP protokole buvo išskirtas papildomų parametrų laukas, kuriame galima perduoti norimus parametrus. Maksimalus TCP antraštės ilgis negali viršyti 60 baitų, iš kurių 40 baitų gali būti išskirti papildomiems TCP parametrų laukams. Minimalus parametro ilgis turi būti bent 8 bitai (1 baitas). Jei mūsų naujam parametrui reikia 4 bitų, tai dar 4 bitai bus palikti tušti.

Dažniausiai naudojami papildomi TCP parametrai yra šie:

1. MMS parametras, kuris nurodo maksimalų TCP segmento dydį, sesijos sudarymo metu;

2. TCP lango daugiklio parametras (angl. *Window scale* – WindowScale), kuris leidžia apeiti TCP lango parametro apribojimus ir naudoti didesnius nei 65 536 baito duomenų gavimo langus;
3. Selektivaus patvirtinimo parametras (angl. *Selective Acknowledgments* – SACK) paspartina duomenų persiuntimą jei pakeliui dingsta keli paketai. Tokiu atveju siuntėjas persiunčia ne visus paketus nuo prarastojo, bet tik tuos kurie nepasiekė adresato;
4. Tinklo vėlinimo parametras (angl. *Timestamps*) – skirtas apskaičiuoti vėlinimą, kuris atsiranda atliekant TCP duomenų perdavimą;
5. Galimi ir kiti, nestandartizuoti TCP parametrai, kurios gali nustatyti duomenų siuntėjas ar gavėjas pagal susitarimą.

3.TCP DUOMENŲ PERDAVIMO KONTROLĖ

Šiame skyriuje bus trumpai aptarti TCP protokolo veikimo mechanizmai ir algoritmai, kurie kontroliuoja duomenų išsiuntimo spartą, atlieka duomenų persiuntimą bei sesijos atstatymą po paketų praradimo. Bus parodyti pagrindiniai trūkumai ir problemos, su kuriomis susiduria TCP protokolas bandydamas optimaliai išnaudoti duomenų kanalą.

3.1. TCP duomenų gavimo ir siuntimo langai (CWIN, WIN)

Kiekvienos naujos TCP sesijos pradžioje duomenų siuntėjas ir gavėjas anonsuoja savo maksimalų duomenų priėmimo langą – WIN. Jis nurodo kiek maksimaliai atminties išteklių yra išskirta gaunamiems TCP duomenims. Tačiau WIN parametras nereguliuoja kokiu greičiu duomenys patenka į tinklą, kas yra labai svarbu, norint neperpildyti duomenų perdavimo kanalą.

Nustatant TCP langą, mes pasakome duomenų siuntėjui, kad jis gali maksimaliai išsiųsti tam tikrą kiekį duomenų, pavyzdžiui TCP WIN = 64 KB. Tačiau TCP nieko nežino apie duomenų perdavimo kanalą, tokiu būdu siuntėjas gali būti pajungtas per 100 Mbit/s duomenų perdavimo spartos liniją, o duomenų gavėjas tik 1 Mbit/s. Jei visus 64 KB duomenų TCP siuntėjas išsiųstų per 0.01 s, tai duomenų perdavimo sparta būtų 64 Mbit/s. Tai daugiau nei 6 kartus viršija TCP duomenis priimančio įrenginio linijos spartą, dėl ko gali atsirasti duomenų praradimai.

Tam tikslui TCP mechanizmas papildomai reguliuoja duomenų išsiuntimo greitį su CNWD parametru. CNWD dar kitaip žinomas kaip šaltinio (siuntėjo) pusės langas arba persipildymo langas. Jis nurodo kiek duomenų siuntėjas gali išsiųsti negavęs patvirtinimo paketo. Nors jis yra labai panašus į WIN, tačiau nereikia jų maišyti:

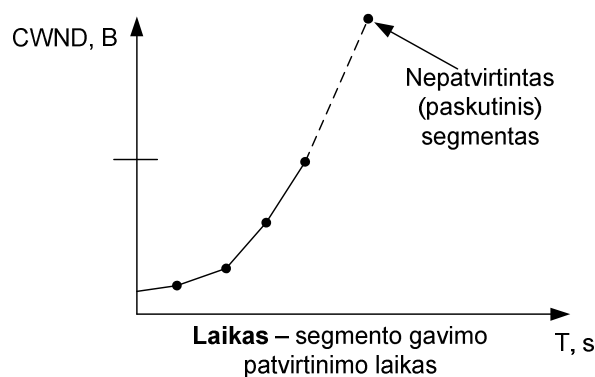
1. WIN – informuojame duomenų siuntėją TCP pakete, kiek maksimaliai mes galime priimti duomenų. Tai nurodo priimančios sistemos TCP atminties (buferio) kiekį;
2. CWIN – nurodo kiek duomenų galime išsiųsti negavus patvirtinimo. Naujos sesijos metu abu TCP sesijos nariai nurodo savo WIN langus. Tačiau CWIN parametras yra nustatomas daug mažesnis už WIN reikšmę. Sesijos metu CWIN reikšmė palaipsniui didinama po kiekvieno gauto patvirtinimo.

Naudodami CWIN parametą galima reguliuoti išsiunčiamų duomenų greitį, kuris tiesiogiai priklauso nuo gaunamų patvirtinimų, tačiau būtina žinoti, kad CWIN parametras negali viršyti WIN vertės.

3.2. Lėto starto fazė

Kiekvienos naujos TCP sesijos pradžioje duomenų siuntėjas nieko nežino apie duomenų perdavimo liniją, todėl jis išsiunčia vieną TCP segmentą ir laukia TCP ACK paketo. Gavęs jį duomenų siuntėjas žino, kad išsiųstas segmentas pasiekė adresatą ir jis išsiunčia du kitus duomenų segmentus. Jei vėl sulaukiamas TCP ACK paketas, išsiunčiamų duomenų segmentų skaičius yra didinamas iki keturių. Tokiu būdu CNWD lango dydis didėja eksponentiniu dėsnio 2^n .

Toks duomenų išsiuntimo lango didinimas yra žinomas kaip lėto starto algoritmas (angl. *Slow Start – SlowStart*) (6 pav.).



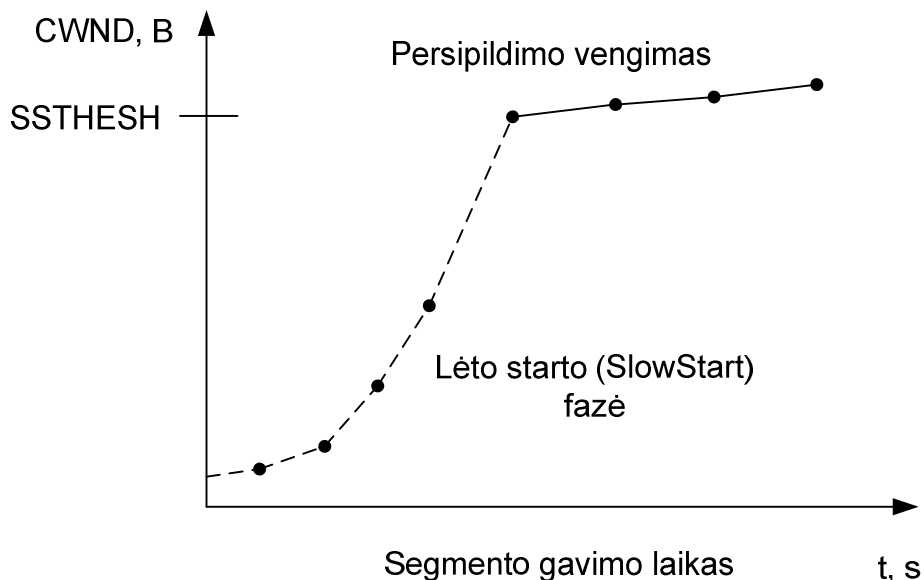
6 pav. TCP lėto starto fazė

Taip didinant duomenų išsiuntimo langą yra pasiekiami maksimali, leidžiama lango riba arba perpildomas duomenų perdavimo kanalas. Po linijos perpildymo pernelyg padidėjęs CNWD yra sumažinamas perpus, o gautoji reikšmė nustatoma kaip slenkstinė (angl. *Slow Start Threshold – Ssthresh*) [2–4]. Jei sumažinta CNWD reikšmė yra mažesnė nei du TCP segmentai, tai Ssthresh yra nustatomas dviem segmentams.

3.3. Persipildymo vengimo fazė

Įvykus linijos perpildymui TCP protokolas pereina į persipildymo vengimo fazę. Duomenų siuntėjas norėdamas išvengti naujų paketų praradimų, pastoviai didina CWIN langą, tačiau tai daro daug atsargiau. Kiekvieną kartą gautoji CWIN reikšmė yra lyginama su Ssthresh verte pagal algoritmą:

1. Jei CNWD yra mažesnis arba lygus Ssthresh reikšmei, tai tada esame lėto starto fazėje, ir tokiu būdu galime padidinti CNWD du kartus (priklauso nuo TCP CWIN valdymo algoritmo);
2. Jei CNWD jau didesnis už Ssthresh, tai padidinus CNWD langą du kartus, tikriausiai tai privestų prie tinklo persipildymo ir duomenų praradimo. Šios stadijos metu CNWD langas yra didinamas mažomis reikšmėmis norint išvengti persipildymo.



7 pav. TCP persipildymo vengimo fazė

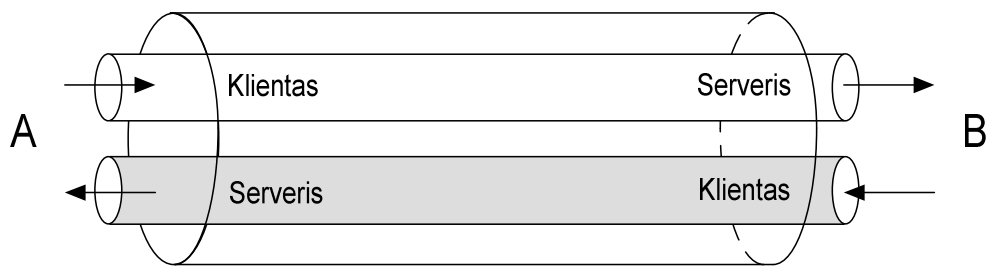
Tokiu būdu veikiant persipildymo vengimo funkcijai, yra pastoviai didindamas CNWD parametras, norint rasti maksimalų perdavimo langą. TCP perkrovos kontrolė buvo sukurta ne tik norint išvengti duomenų perdavimo linijos perpildymo, bet ir norint užtikrinti teisingą ir efektyvų duomenų srauto padalinimą tarp kelių vartotojų bei sesijų [2, 4, 7].

3.4. Tinklo vėlinimo įtaka TCP veikimui

Kaip žinome TCP sesijos sudarymo metu priimta turėti kliento ir serverio pusę (klientas – kuris jungiasi ir siunčia duomenis, serveris – kuris priima duomenis). Tačiau praktikoje toks išsireiškimas nėra visiškai tinkamas ir gali klaidinti, nes ne visada klientas siunčia duomenis. TCP sesijos metu serveris gali tiek siųsti tiek priimti duomenis. Siekdami patogumo mes naudosime A ir B išsireiškimą, kur A ir B turi savyje klientą ir serverį (11 pav.).

Taigi sesijos sudarymo metu TCP sudaro virtualų (angl. *Full-duplex*) duomenų kanalą tarp A ir B. Kanalo sudarymo metu A serveris nurodo savo lango dydį WIN, kuris nusako kiek maksimaliai duomenų gali būti tinkle be patvirtinimo ACK, kitaip sakant kiek duomenų gali priimti serveris vienu metu. Tokį pat veiksmą atlieka ir B serveris, kuris irgi nurodo savo lango WIN dydį (A ir B serverių langai WIN gali skirtis).

Jei A klientas nori siųsti duomenis, jis turi atsižvelgti į B serverio nurodytą langą ir negali jo viršyti (siuntėjo pusės langas turi būti visada mažesnis (arba lygus) gavėjo nurodytam langui $WIN \geq CNWD$). Tokia pati taisyklė galioja ir B pusės klientui.



8 pav. TCP loginis perdavimo kanalas

Tipinės TCP sesijos sudarymo metu WIN langas parenkamas santykinai mažas ir gali būti per mažas greitam duomenų persiuntimui. Tokiu atveju TCP sesijos metu A ir B pusės pastoviai informuoja apie savo lango WIN dydžio pokyčius, tokiu būdu leidamos optimaliai adaptuotis prie sisteminių ir tinklo išteklių.

Tačiau TCP lango dydis (WIN) gali kisti tik nuo 0 iki 65 535 B ribose, nes TCP antraštėje yra išskirta tik 16 bitų ($2^{16} = 65\,535$ B). Norint naudoti didesnę langą nei 65 535 B, yra naudojamas WindowScale TCP parametras, dėl kuri lango dydis gali siekti iki 1 GB dydžio. Naudojant šį parametą naujasis langas WINs yra gaunamas dauginant senąjį WIN iš 2^n . Kur n yra WindowScale vertė nuo 0 – 14.

$$WIN_s = WIN \cdot 2^n, \quad (3.1)$$

TCP WindowScale parametras leidžia apeiti WIN parametro apribojimą ir pasiekti maksimalią 1 GB lango dydį, naudojant maksimalią reikšmę 14 (jei parametras nustatomas ties nuli, tai WindowScale yra nenaudojamas). Šis parametras yra nustatomas naujos sesijos sudarymo metu ir negali būti keičiamas TCP sesijos metu.

Nors WIN parametras nurodo priimančio kliento TCP atminties kiekį (buferį), tačiau jis tiesiogiai daro įtaką CWIN parametrai, kuris turi būti mažesnis arba lygi WIN reikšmei.

Nustačius TCP WIN parametrai per mažą, turėsime lėtą duomenų perdavimo greitį. Nustatę per didelį – sisteminiai OS ištekliai bus panaudoti ne optimaliai ir atsiras papildomas vėlinimas. Optimalus WIN (CWIN) lango dydis turi būti lygus BDP (angl. *Bandwidth Delay Product*) reikšmei [6]. BDP labai panašus į WIN, tik jis nusako kiek maksimaliai duomenų gali būti tinkle (tranzite) tarp siuntėjo ir gavėjo. Dažniausiai BDP yra didesnis už WIN ir todėl yra neefektyviai išnaudojamas duomenų perdavimo kanalas.

Norint apskaičiuoti BDP parametrai, reikia žinoti siauriausią duomenų kanalo pralaidą ir bendrą vėlinimą tarp TCP kliento ir serverio.

$$BDP(\text{bitai}) = B(\text{bitai} / \text{s}) \cdot RTT(\text{s}), \quad (3.2)$$

arba

$$BDP(Kbaitai) = B(Kbaitai / s) \cdot RTT(ms), \quad (3.3)$$

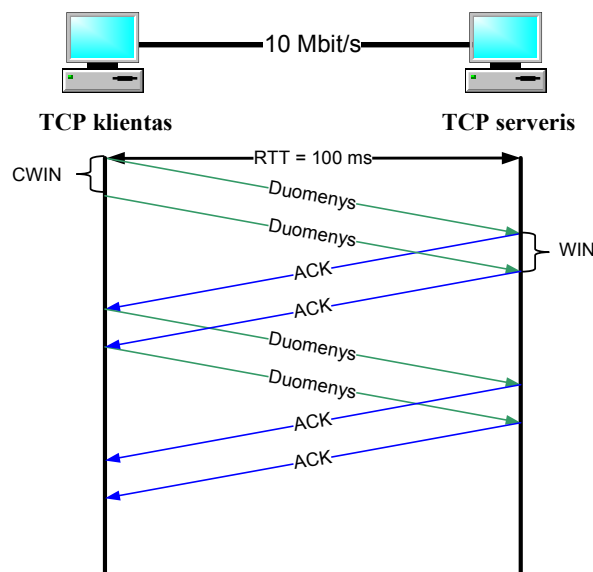
čia BDP – nusako linijos talpą, b; B – linijos pralaidumas, b/s; RTT – linijos vėlinimas į abi puses;

Tarkime turime du kompiuterius sujungtus 10 Mbit/s linija. Vėlinimas į vieną pusę siekia ~50 ms (RTT = 100 ms). Abi pusės susitarė dėl TCP lango WIN = 62,5 KB dydžio reikšmės. Tai reiškia, kad TCP siuntėjo A maksimalus išsiuntimo langas (CWIN) gali būti 62,5 KB, tai yra duomenų kiekis, kuris gali būti išsiųstas negavus ACK patvirtinimo.

Kadangi fizinė linijos pralaida yra lygi 10 Mbit/s, tai A kompiuteriui prireiks ~ 50 ms išsiųsti 62,5 KB duomenų (laikydami, kad CWIN yra maksimalus):

$$t_{WIN} = \frac{WIN}{B} = \frac{62,5 \cdot 10^3 \cdot 8}{10^7} = 5 \text{ ms}, \quad (3.4)$$

Po duomenų išsiuntimo A kompiuteris dar turės laukti 50 ms patvirtinimo (ACK) iš B, kad gautas pirmas paketas ir galima siųsti kitus paketus.

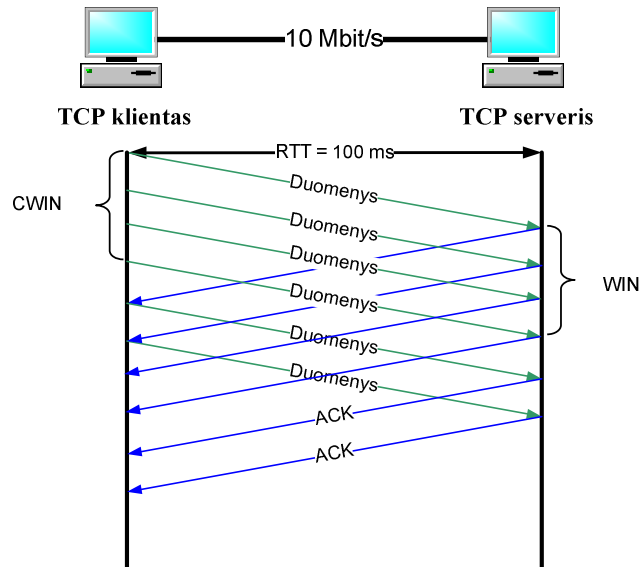


9 pav. Vėlinimo įtaka TCP spartai

Kaip suprantama duomenų perdavimo kanalas nėra efektyviai išnaudojamas, o vidutinis duomenų perdavimo greitis tesiekia $B = 5 \text{ Mbit/s}$ (9 pav.). Esant tokiai apkrovai, duomenų perdavimo kanale pastoviai bus 62,5 KB duomenų, kai maksimaliai ten galėtų būti $BDP = 125 \text{ KB}$.

$$BDP = B \cdot RTT = 10 \cdot 10^6 \cdot 100 \cdot 10^{-3} = 10^5 = 125 \cdot 10^3 \text{ B}, \quad (3.5)$$

Norėdami pasiekti optimalią duomenų perdavimo spartą, TCP lango dydis turi atitikti BDP dydžiui $WIN = DP = 125\,000$ baitai. Siuntėjui A prireiks apytiksliai 100 ms tam, kas išsiųsti 125 KB duomenų. Per tą laiką iš duomenų gavėjo turėtų ateiti pirmas patvirtinimas, kad gautas pirmas iš išsiųstų paketų ir galima siųsti kitus (10 pav.).



10 pav. Optimalus TCP išnaudojimo pavyzdys

Norint pasiekti optimalų linijos išnaudojimą reikia, kad WIN reikšmė būtų lygi ar didesnė už BDP reikšmę. Tačiau daugelis OS standartiškai WIN reikšmė neviršija 64 KB vertės.

Nors tai gali pasirodyti gan mažai, tačiau to pilnai užtenka 100 Mbit/s LAN atveju, kuomet RTT neviršija 5 ms ir yra pilnai išnaudojamas linijos pralaidumas. Apytiksliai naudojamą TCP lango (WIN) dydį galima rasti žinant RTT laiką ir duomenų perdavimo greitį.

$$WIN = RTT \cdot B_{\text{prakt}}, \quad (3.6)$$

WIN – gavimo lango dydis, B; RTT – vėlinimas, į abi puses, s; B_{prakt} – linijos pralaidumas, B/s;

Kai $RTT = 5$ ms, tai norint visiškai išnaudoti 100 Mbit/s duomenų kanalą TCP langas (WIN) turės būti:

$$WIN = 100 \cdot 10^6 \cdot 5 \cdot 10^{-3} = 5 \cdot 10^5 (\text{baitai}) = 62,5 \text{ KB}, \quad (3.7)$$

Nors kaip matyti 64 KB WIN lango dydžio pilnai užtenka daugeliu LAN atveju, tačiau linijose su dideliu vėlinimu (angl. *Long Fat Networks* – LFN) ar WAN su didesniu vėlinimu, 64 KB TCP lango reikšmės nepakanka.

Nors perduodamų duomenų greitis tiesiogiai priklauso nuo TCP WIN parametro ir optimalus dydis turėtų būti lygus BDP, tačiau praktikoje WIN turėtų būti šiek tiek didesnis nei BDP. Mūsų

pavyzdyje pateiktas tinklas ir TCP sesijos nariai buvo tik teoriniai, neįvertinantys vėlinimo ir nekintantys laike. Praktikoje to niekada nebūna.

3.5. Pakartotinio persiuntimo laikas (RTO)

TCP sesijos metu sėkmingą duomenų perdavimą parodo gautas ACK iš TCP kliento (10 pav.). Įvykus duomenų praradimui ir dingus vienam ar keliems TCP duomenų paketams, klientas (duomenų gavėjas) informuoja serverį išsiųsdamas tris ar daugiau identiškų ACK paketų, po kurių yra duomenų persiuntimas iš serverio. Nors šis mechanizmas veikia gerai daugeliu atvejų, tačiau tam tikruose asimetrinėse perdavimo linijose su dideliu paketų praradimu galimos problemos.

Taip gali nutikti, kai duomenų perdavimo linijoje paketų praradimai vyksta tik į vieną pusę ir dingus vienam TCP duomenų paketui, visi trys (ar daugiau) ACK paketai nepasiekia serverio. TCP serveris negavęs informacijos apie paketų praradimus, paprasčiausiai nustoja siųsti duomenis ir pereina į duomenų laukimo stadiją.

Kad taip nenutiktų, TCP mechanizme yra pakartotinis persiuntimo laikmatis (angl. *Retransmission Timer* – RTO), per kurį turi būti gautas ACK paketas. Jei per RTO laiką nėra gaunamas ACK paketas laikoma, kad paskutinis TCP duomenų paketas (kuriam nebuvo gautas patvirtinimas) yra dingęs ir įvykdomas jo persiuntimas.

Dėl skirtingų tinklo vėlinimų ir deviacijų RTO laikas yra dinaminis ir paskaičiuotas iš RTT laiko:

$$t_{\text{RTO}} = \overline{t_{\text{RTT}}} + 4 \cdot \sigma_{\text{RTT}}, \quad (3.8)$$

čia $\overline{t_{\text{RTT}}}$ – vėlinimo vidurkis; σ_{RTT} – vėlinimo deviacijos kvadratinė šaknis. Gautoji vertė t_{RTO} yra laikoma ribine, per kurią turi būti gautas ACK paketas.

4.TCP LANGO PERSIPILDYMO VENGIMO ALGORITMAI

Šiame skyriuje bus trumpai aptarti TCP protokolo persipildymo vengimo algoritmai, kurie kontroliuoja duomenų išsiuntimo greitį.

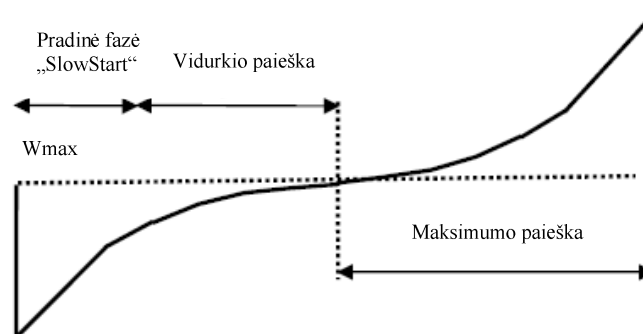
4.1. TCP CUBIC algoritmas

Dauguma esamų ir naujai sukurtų TCP protokolo variacijų daugiausia dėmesio skiria lango funkcijos (CNWD) augimui pagerinti, tačiau išlieka viena problema, tai yra tolygus srauto padalinimas esamiems naujiems TCP srautams su skirtingais RTT laikais.

CUBIC TCP protokolas yra BIC TCP protokolo patobulinimas, kuris supaprastina CNWD kontrolės algoritmą ir yra daug palankesnis kitiems TCP srautams.

BIC lango didėjimo funkcija

Norint perprasti CUBIC veikimo algoritmą iš pradžių reikia susipažinti su jo pirmtako BIC veikimo principu. Žemiau paveikslėlyje pavaizduota TCP BIC lango didėjimo funkcija.



11 pav. BIC lango augimo funkcija

Kada linijoje atsiranda paketų praradimai BIC algoritmas sumažina lango dydį CNWD daugindamas iš β . Tokiu būdu sumažinamas CNWD lango dydis nuo W_{max} iki W_{min} . Po to kai langas buvo sumažintas iki W_{min} , BIC algoritmas atlieka optimalaus lango paiešką [3]. Naujo lango dydis bus vidurkis tarp W_{max} ir W_{min} reikšmių. Jei langų skirtumas tarp W_{min} ir W_{max} yra gana didelis, tai parenkant tarpinę reikšmę tarp minimo ir maksimumo gali būti per didelis lango prieaugis [7]. Norint to išvengti žiūrima, kad skirtumas tarp vidurio reikšmės ir W_{min} nebūtų didesnis už S_{max} konstantą. Jei taip atsitinka:

$$\frac{W_{max} - W_{min}}{2} > S_{max} , \quad (4.1)$$

tokiu atveju TCP lango augimo funkcija pradžioje bus linijinė (lėto augimo dalis), po kurios bus logaritminė dalis (vidurkio paieškos dalis).

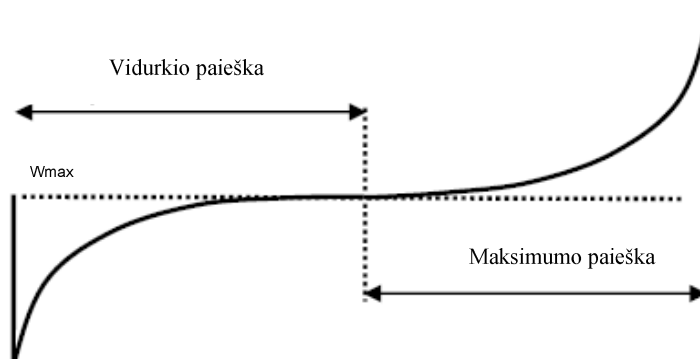
Jei lango dydis viršija prieš tai buvusį, tai reiškia, kad lango dydis yra didesnis nei nustatytasis ir būtina rasti naują didžiausią lango reikšmę. Tada BIC pereina į naujo maksimumo paieškos režimą (angl. *max probing*). Šis etapas yra priešingas prieš tai buvusiam [7]. Iš pradžių lango dydis didėja pagal eksponentinę (kuri yra lėta pradžioje) priklausomybę, kurio metu bandomas surasti naujas maksimumas. Jei po šio etapo nebuvo rastas naujas W_{max} , tai nauja reikšmė yra gaunama dauginant esamą lango dydį iš tam tikros konstantos (tiesinė funkcija). Kaip matyti čia naudojama tas pats maksimalaus lango nustatymo algoritmas po sumažėjimo tik kita tvarka.

Tokiu būdu naudojant TCP BIC algoritmą nustatomi tokie dėsningumai:

1. Teisingas ir tolygus TCP srauto pasidalinimas tarpusavyje, kai CNWD artėja prie maksimumo;
2. Artėjimas prie maksimalaus lango dydžio yra labai lėtas, todėl nedaroma didelė įtaka gretimiems srautams (nesukelia paketų praradimų gretimuose srautuose);
3. Šiuo principu pasiekiamas maksimalus tinklo išnaudojimas prie W_{max} reikšmės;
4. Pasiekiamas tolygus srauto pasidalinimas, kai tinkle yra keli srautai su skirtingais RTT.

CUBIC lango didėjimo funkcija

Nors TCP BIC lango nustatymo funkcija yra gana gera ir palanki kitiems TCP srautams, bet ji dažniausiai taikoma tinklams su dideliu duomenų pralaidumu. Tinkluose, kuriuose duomenų kanalo pralaidumas yra palyginti nedidelis, o RTT laikai mažas, BIC funkcija gali elgtis gana agresyviai kitų TCP srautų atžvilgiu. Norint išspręsti esamas problemas ir dar labiau supaprastinti BIC protokolo veikimo algoritmą buvo pasiūlytas patobulinimas CUBIC. Naujoje BIC versijoje buvo tai pat patobulinta lango augimo funkcija (12 pav.).



12 pav. CUBIC lango didėjimo funkcija

TCP CUBIC funkcijoje lango augimas po sumažinimo yra daug didesnis nei BIC ir sulėtėja ties buvusiu $W_{\max}$ reikšme (ekstremumu), kurioje lango prieaugis pasidaro lygus nuliui [3]. Po to kai buvęs lango dydis $W_{\max}$ buvo viršytas, funkcija vėl pradeda didėti. Šis etapas yra priešingas prieš tai buvusiam. Iš pradžių lango dydis didėja pagal eksponentinę (kuri didėja lėtai funkcijos pradžioje) priklausomybę, kurio metu bandomas surasti naujas maksimumas. Po to pereina į linijinį augimą.

Kaip matyti iš 6 pav. CUBIC funkcija kinta daug lėčiau ties ekstremumu nei BIC, dėl to gaunamas tolygesnis duomenų srautas.

TCP CUBIC lango kitimo grafiką nusako tokia funkcija:

$$K = \sqrt[3]{W_{\max} \frac{\beta}{C}}, \quad (4.2)$$

β – konstanta, CNWD lango mažinimo faktorius, kuris nusako kiek lango dydis turi būti sumažintas jei atsiranda paketų praradimai ($\beta W_{\max}$).

Iš 6 pav. matyti, kad CNWD lango augimo funkcija po sumažinimo auga greitai, tačiau priartėjusi prie $W_{\max}$ sulėtėja ir ties $W_{\max}$ prilygta nuliui. Po to kai buvo viršyta $W_{\max}$ reikšmė CNWD lango augimas pradeda iš po lėto greitėti tolygiai nuo $W_{\max}$ [7].

TCP CUBIC taip pat užtikrina tolygų kanalo pasidalinimą tarp konkuruojančių srautų. Jei duomenų kanale, kuriame yra du skirtingi TCP srautai įvyksta perpildymas, tai abu jie sumažina savo lango dydį β daugiklio ($\beta W_{\max}$). Tokiu būdu pagal 9 formulę TCP srautas su didesniu langu bus sumažintas daugiau ($\beta W_{\max}$) nei su mažesniu CNWD, o po sumažinimo TCP srauto su didesniu $W_{\max}$ augimas bus lėtesnis (nes K su didesniu $W_{\max}$ bus didesnis) nei TCP srauto su mažesniu $W_{\max}$. Tokiu būdu kanalo pasidalinimas tarp skirtingų TCP srautų palaipsniui susilygins.

5. ACK MECHANIZMAS

Pagrindinė TCP protokolo paskirtis yra patikimas duomenų perdavimas, nesvarbu ar tai būtų nutolę tinklo mazgai ar duomenų apsikeitimas tarp vidinių kompiuterio programų. Tam užtikrinti TCP protokolas naudoja mažus TCP ACK patvirtinimo paketus, kurie ne tik informuoja duomenų siuntėją apie sėkmingai perduotus duomenis, bet ir apie paradimus. Dažniausiai tokiuose TCP paketuose nebūna jokio papildomo duomenų lauko ir jie neperneša jokių papildomų duomenų išskyrus tarnybinius. Kartais ši informacija yra nereikalinga ir ją galima laikyti tiesiog pertekline. O kas yra nereikalinga, tas gali būti pašalinta.

Šiame skyriuje bus pateiktas ir apžvelgtas TCP ACK patvirtinimo mechanizmo veikimo pavyzdys, apžvelgti TCP ACK pertekliaus mažinimo būdai, jų privalumai bei trūkumai.

5.1. TCP ACK patvirtinimo mechanizmas

Kaip jau buvo aptarta pirmoje darbo dalyje, TCP ACK pranešime perduoda savo SEQ ir ACKn numerius, kurie patvirtina prieš tai gautus duomenis. Jei TCP ACK paketas neperduoda papildomų duomenų ir nėra nustatyta papildomų parametrų išskyrus ACK, tai SEQ numeris nekinta ir išlieka toks pat. Tai nutinka, kai duomenų apsikeitimas vyksta viena kryptimi – iš serverio klientui arba atgal.

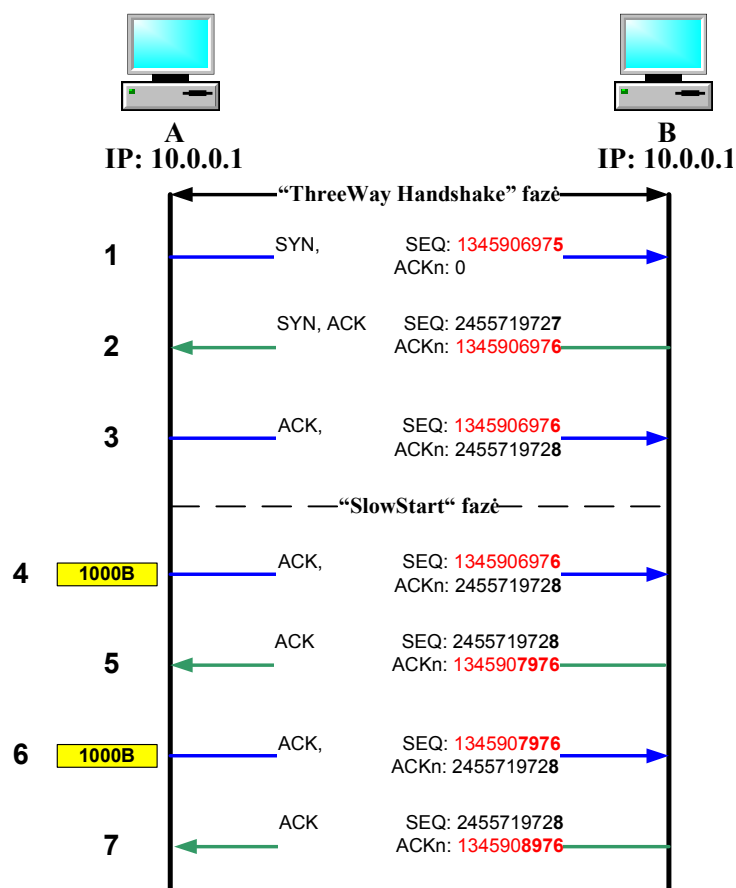
Prieš pradedant analizuoti pavyzdį, reikėtų paminėti, kad TCP dar turi požymių lauką (angl. *Options fields*), kuriame yra SYN, ACK, PSH ir kiti požymiai. Jie yra naudojami kiekvienos naujos sesijos sudarymo, eigos ir nutraukimo metu [4]. Tarp šių parametrų yra SYN ir ACK laukai, kurių nereikėtų maišyti su eiliškumo (SEQ) ir patvirtinimo (ACKn) laukais, kurie atsakingi už paketų eiliškumą bei duomenų patvirtinimą.

Panagrinėkime vienos TCP sesijos sudarymą ir eigos procesą, ir kaip tuo metu kinta (SEQ) ir patvirtinimo (ACKn) laukai. Dėl aiškumo tarkime, kad klientas (A) jungiasi į serverį (B) ir siunčia duomenis.

1. Kiekviena nauja sesija prasideda nuo trijų dalių sesijos sudarymo mechanizmo – ThreeWay Handshake. Klientas norėdamas sudaryti sesiją su serveriu, siunčia TCP paketą su nustatytu SYN parametru bei unikaliu eiliškumo SEQ $N = 1345906975$ numeriu. Patvirtinimo ACKn laukas yra lygus nuliui – tuščias (13 pav.).
2. TCP serveris atsako paketu su nustatytais SYN ir ACK parametrais ir yra sugeneruojamas antras, unikalus eiliškumo SEQ numeris $N2 = 2455719727$. Šiuo paketu yra patvirtinama, kad gautas prieš tai išsiųstą paketą. Patvirtinimo lauke yra įrašomas siuntėjo eiliškumo

numerį plus 1 – $ACK_n = N+1 = 1345906976$. Taip TCP serverį informuoja, kad gautas prieš tai siųstas paketas, kito paketo eiliškumo numeris turėtų būti $N+1$.

3. Klientas gavęs šį paketą atsako TCP paketu su ACK parametru bei padidina savo eiliškumo numerį per vieną: $SEQ\ N2+1 = 1345906976$ (kuris turi būti lygus prieš tai iš serverio gautam paketo patvirtinimo numeriui). Kliento patvirtinimo numeris padidinimas per vieną, $ACK_n = 2455719728$. Taip serveris informuojamas, kad jo siųstas paketas yra gautas ir tikimasi, kad kitas gautas paketas turės $SEQ = N+1$ dydį.



13 pav. pav. TCP patvirtinimo mechanizmas

4. Pasibaigus ThreeWayHandshake, TCP sesija pereina į SlowStart. Kaip matyti 13 pav., klientas į serverį dar išsiunčia 1000 baitų duomenų paketą, tačiau SEQ ir ACK_n nepasikeitė nuo 3 punkto, nes nebuvo gautas naujas TCP ACK paketas iš serverio. Kaip jau buvo minėta, siunčiamo paketo eiliškumo numeris – SEQ turėtų atitikti prieš tai gauto patvirtinimo numeriui – 1345906976 . Eiliškumo numeris išlieka tas pats, nes klientas dar negavo paketo iš serverio su $N2 = 2455719728$.
5. Serveris gavęs TCP paketą atsako, kad paketas gautas ir tikimasi, kad kitas paketas turės vertę $SEQ\ N = 1345907976$. Pasibaigus ThreeWayHandshake fazei, ACK_n patvirtina ne paketo

eiliškumą, o duomenis $ACK_n = 1345906976 + 1000$. Kiekvienas iš TCP sesijos narių yra sėkmingai gavęs duomenų – baitų.

6. Klientui išsiuntus dar 1000 baitų duomenų. SEQ tampa. 1345907976., Tačiau kad ACK_n liko toks pat kaip ir 5 pakete 2455719728. Tai įvyksta todėl, kad siunčiamas TCP paketas neturi papildomų duomenų ir yra nustatytas tik ACK parametras.

7. Serveris atsako TCP ACK paketu ir padidina ACK_n lauką iki 1345908976 palikdamas SEQ nepakitusį.

Didžioji TCP sesijų dalis yra vienkrypčiai, kai duomenys yra perduodami iš kliento į serverį arba antraip. Tokios sesijos generuoja didelį kiekį ACK paketų, kurių naudingoji dalis tesudaro 20 % arba 4 baitus pranešimo (paketo). Žinant, kad TCP yra naudojamas kartu su transportavimo ir IP protokolais, tai naudinga informatyvioji paketo dalis tipiniuose Ethernet tinkluose nesudaro net 7 %. Nors TCP ACK paketas turi didelį duomenų perteklių, tačiau jie yra gyvybiškai svarbūs sesijos stabilumui bei duomenų perdavimo patvirtinimui.

5.2. ACK pertekliaus mažinimas

Didelis perteklius, kurį generuoja TCP ACK paketai, kai kuriose aplinkose turi neigiamą poveikį TCP sesijos spartai ir stabilumui. Tai pastebima asimetriniuose duomenų perdavimo linijose, kur siauras duomenų išsiuntimo/gavimo kanalas gali būti pilnai perpildytas ACK paketais arba duomenų perdavimo kanaluose, kur duomenų išsiuntimui ir gavimui yra naudojamas tas pats fizinis kanalas. Didelis ACK paketų skaičius taip pat neigiamai veikia tinklo įrenginius sukurdamas papildomą apkrovą adoruojant gautus duomenų paketus.

Norint sumažinti ACK paketų duomenų perteklių ir pagerinti TCP sesijos veikimą buvo pasiūlyta keleta būdų:

1. ACK kompresija;
2. Delayed ACK algoritmas;
3. ACK filtravimas.

5.3. ACK kompresija

Kaip jau buvo minėtą anksčiau TCP ACK paketai turi labai didelį perteklių, kurie turi neigiamą įtaką TCP spartai. Tai labai pastebima didelio vėlinimo palydoviniuose ryšio linijose, kai duomenų gavimui yra naudojamas didelės spartos palydovinis ryšys, o išsiuntimui mažo greičio modelinis ryšys. Dažnai tokiose sistemose didelis skaičius ACK paketų pilnai apkrauna išsiuntimo liniją ir neleidžia pasiekti didesnio parsisiuntimo greičio vien dėl mažo ACK paketų skaičiaus.

Norint sumažinti TCP ACK pertekliu buvo pasiūlyta TCP ir IP protokolo antraščių kompresija, kuri sumažina paketo dydį net 75 %. Atlikus kompresiją ne tik atlaisvinamas išsiuntimo kanalas, bet ir

leidžiamas didesnis ACK paketų skaičius į serverį, kuris tiesiogiai daro įtaką duomenų parsisiuntimo greičiui bei sesijos atsistatymą po paketų praradimo.

Norint atlikti TCP ACK kompresiją reikalinga, kad būtų įdiegti tarpiniai tinklo mazgai. Tai yra vienas ir didžiausių trūkumų, su kuo susiduria ši technologija, nes tai nereikalauja papildomų įrenginių ar sisteminių išteklių.

5.4. Pavėlintas ACK (Delayed ACK)

Kitas TCP ACK mažinimo būdas remiasi tuo, kad TCP ACK paketas gali būti generuojamas daug rečiau nei kiekvienam antram pilnam TCP duomenų segmentui [8]. Norint tai atlikti, reikalingas programinio kodo pakeitimas TCP programoje ar sistemoje. Nors šis metodas yra labai lankstus ir turi didelį potencialą, tačiau tai turi palaikyti abu TCP sesijos nariai, nes sumažintas TCP ACK skaičius mes sumažiname TCP sesijos spartą ir atsistatymo, po paketų praradimų. Didelis šio metodo trūkumas yra tai, kad kodo pakeitimai turi būti atlikti visuose programose ar OS sistemose, kas yra labai sudėtinga ir pratiškai neįmanoma.

5.5. ACK filtravimas

Vienas iš galimų TCP ACK paketų mažinimo būdų yra naudoti ACK filtravimą, kuris pašalina dalį ACK paketų iš bendro srauto. Toks sprendimas remiasi kumuliaciniu ACK principu, kai paskutinis ACK paketas patvirtina prieš tai visus gautus duomenis. Šis metodas gerai žinomas ir naudojamas praktikoje XDSL ir bevielio ryšio perdavimo sistemose [8–10], kur yra asimetrinės perdavimo linijos.

Kadangi ACK paketai be duomenų patvirtinimo dar dalyvauja CWIN lango augimo procese bei paketų persiuntime, tai ACK filtravimas įmanomas tik nedidelio vėlinio tinkluose, kur jis neturi didelės įtakos TCP sesijos spartai ir stabilumui.

Kadangi dauguma tinklo įrenginių naudoja buferius gaunamiems ir išsiunčiamiems paketams, tai ACK filtravimas gali būti atliekamas paketų buferyje iš jo pašalinant seniausius ACK paketus.

Didelis ACK filtravimo privalumas yra tai, kad jis gali būti atliekamas tarpiniuose tinklo mazguose ir yra palyginti lengvai įgyvendinamas, nereikalaujantis sudėtingų duomenų apdorojimų. Tačiau atliekant filtravimą yra prarandami duomenys ir tai gali įtakoti TCP sesijos spartos augimą bei stabilumą.

Sparčiai didėjant duomenų perdavimo spartoms bei technologijoms vis didesnė apkrova tenka tinklo įrenginiams, kuriems per sekundę tenka apdoroti šimtus tūkstančių paketų ar net daugiau. Didesniąją tinkle esančių paketų dalį sudaro TCP/IP paketai, kurie garantuoja patikimą duomenų perdavimą iš vieno tinklo įrenginio į kitą. Nors šis protokolas pasižymi patikimumu, tačiau jis turi didelį papildomų duomenų perteklių, kuris yra būtinas lėtuose perdavimo linijose, kur yra dažnas

duomenų praradimas, tačiau plačiąjuosčiuose duomenų kanaluose su mažu paketų praradimu šis TCP protokolo perteklius yra visiškai nereikalingas.

6. ACK FILTRAVIMO MECHANIZMAI

Kaip jau buvo minėta, TCP ACK protokolo informacijos perteklių galima mažinti keliais būdais. Vienas iš jų yra ACK filtravimas, kuris yra paprasčiausias ir nereikalaujantis sudėtingų tinklo ar galinių įrenginių pakeitimo. Toks ACK filtravimas yra gerai žinomas ir aptartas daugelyje straipsnių [8–11]. Tačiau kaip ir visi pasiūlyti ACK informacijos perteklių mažinantys sprendimai turi savo trūkumus: didina vėlinimą ir sulėtina TCP sesijos augimą.

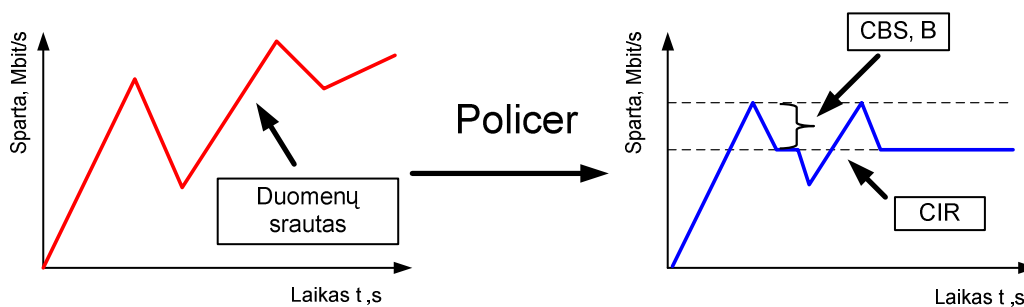
Norint sumažinti vėlinimą ir apkrovą tenkančią įrenginiams atliekant TCP ACK filtravimą darbo metu buvo naudojamas nustatytų sąlygų (angl. *policer*) ACK filtravimo mechanizmas, kuris pasižymi geresnėmis vėlinimo savybėmis ir yra tinkamas apdoroti didelius kiekius ACK paketų.

Šiame skyriuje bus atlikti praktiniai matavimai norint surasti optimalų stabilų ACK filtravimo lygį, pateiktos pagrindinės problemos ir ypatumai susiję su TCP ACK filtravimo mechanizmu.

6.1. Nustatytų sąlygų ACK filtravimas

Tipiniai ACK filtravimo mechanizmai yra paremti buferio filtravimu, kai TCP ACK paketas yra trinamas tik tada, kai tinklo sąsajoje atsiranda kitas tos pačios TCP sesijos ACK paketas [8, 9, 11].

Šio darbo metu mes pasiūlėme naują, alternatyvų nustatytų sąlygų (angl. *policer*) ACK filtravimo mechanizmą, kuris geriau tinkamas didelių srautų filtravimui ir jų apdorojimui lyginant su standartiniais ACK filtravimo mechanizmais. Tai buvo realizuota pasinaudojant *tc* linux operacinės programa, kuri atliko paketų klasifikavimą ir filtravimą pagal nustatytus CIR ir CBS parametrus (14 pav.).



14 pav. Policar mechanizmo veikimas

Nustatytų sąlygų filtravimas turi tokius privalumus lyginant su tipiniais ACK filtravimo mechanizmais:

1. Naudojant nustatytų sąlygų ACK filtravimą paketų šalinimas yra vykdomas tik viršijus nustatytą ribinę paketų spartą (CIR). Dėl to TCP sesija yra mažiau paveikta pradinėje – lėto starto fazėje, o leidžiamas laikinas paketų padidėjimas (CBS) leidžia TCP sesijai greičiau atsistatyti po paketų praradimų greito persiuntimo metu;

2. Nustatytų sąlygų filtravimas gali būti atliekamas prieš paketams patenkant į tinklo sisteminę sąsają (angl. *Network Interface Controller* – NIC), tai yra anksčiau negu bus atliekama paketo analizė. Taip yra sutaupomi ne tik sistemos ištekliai, bet ir sumažinamas vėlinimas, todėl padidėja TCP sesijos sparta;
3. Priešingai nei tipiniai ACK filtravimo mechanizmai, šis metodas yra daug paprastesnis ir lengviau įgyvendinamas bei nereikalauja sudėtingų sistemos ar programinio kodo pakeitimų.

Kadangi nustatytų sąlygų ACK filtravimas nagrinėjamas pirmą kartą ir tiksliai nėra žinoma jo įtaka sistemos apkrovai bei sisteminiams ištekliais, tai darbo metu ACK filtravimas buvo atliekamas tarpiniame tinklo įrenginyje. Tam tikslui buvo panaudotas kompiuteryje su Linux operacine sistema, kuris turėjo dvi Ethernet tinklo sąsajas (NIC) ir veikė kaip skaidrus tinklo mazgas. Jis skaidriai, nepakeisdamas per save perduodavo tinklo paketus. Duomenų filtravimas buvo atliekamas tik iš kliento pusės ir filtruojant ACK paketus, kurie neperneša papildomos informacijos. Programinis kodas naudojant *tc* programą [12] pateiktas žemiau:

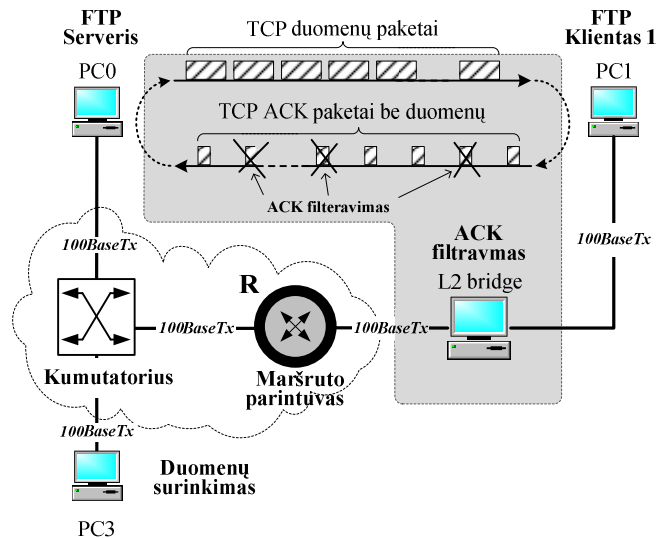
```
tc filter add dev eth1 parent ffff:0 protocol all prio 1 u32 match u32 \
0xaff0001 0xffffffff at 16 classid ffff:0 police index 2 rate 12500bps \
burst 102400 mpu 0 action drop/pass
tc filter add dev eth1 parent ffff:0 protocol all prio 1 u32 match u32 \
0x0 0x0 at 0 classid ffff:0 police index 3 rate 1bps burst 1 action \
drop/drop
```

Papildomai norint sumažinti galimą vėlinimą, bei vėlinimo deviacijos padidėjimą dėl nustatytų sąlygų ACK filtravimo, kompiuterio operacinė sistema buvo nustatyta dirbti didesniu sisteminiu dažniu. Tai padidino sistemos veikimo tikslumą ir sumažino paketų apdorojimo laiką.

Testavimo aplinka

Norint ištirti kaip ACK filtravimas veikia TCP sesiją, jos stabilumą bei spartą buvo atlikti tyrimai realiame tinkle. Testavimo tinklą sudarė du galiniai įrenginiai – kompiuteriai: rinkmenų apsiųtimo (angl. *File Transfare Protocol* – FTP) serveris ir FTP klientas. Antrasis atliko duomenų failo parsisiuntimą iš FTP serverio. Tarp serverio ir kliento buvo sudarytas mažas duomenų perdavimo tinklas, kurį sudarė Alcatel komutatorius ir Cisco tinklo maršruto parinktuvas. FTP serveryje veikė su TCP CUBIC persipildymo vengimo algoritmas.

ACK filtravimas buvo atliekamas tarpiniame kompiuteryje tarp FTP kliento ir Cisco maršruto parinktuvo (15 pav.).



15 pav. Tinklo schema

Perduodamų duomenų stebėjimui ir surinkimui buvo panaudotas komutatoriaus porto kopijavimas (angl. *port monitor*). Komutatoriuje visi gaunami ir išsiunčiami duomenys buvo nukreipiami į duomenų surinkimo kompiuterį. Detali tinklo įrenginių ir kompiuterių konfigūracija pateikta priede.

Norint sumažinti duomenų praradimą ir linijos persipildymą, TCP sparta buvo apribota ant FTP serverio Ethernet tinklo sąsajos pasinaudojus *tc* programa. Ji sukūrė idealų TCP paketų ribotuvą, kuris vėlindavo, bet nedarydavo paketų praradimų [13]. Keičiant *<rate>* parametą buvo reguliuojama duomenų išsiuntimo sparta bandant išvengti TCP paketų praradimų.

```
tc qdisc change dev eth0 root tbf rate 90mbit burst 100kb latency 10ms \
minburst 1540
```

Tokiu būdu visi paketų praradimai ir TCP sesijos nestabilumai galėjo atsirasti tik dėl nustatytų sąlygų ACK filtro veikimo arba maršruto parinktuvo.

6.2. ACK filtravimo įtaka TCP sesijos stabilumui

Norint nustatyti kokia įtaką turi ACK filtravimas TCP duomenų perdavimo spartai ir stabilumui, pirmojo tyrimo metu buvo atliktas žingsninis ACK filtravimo didinimas, filtruojant grįžtančius ACK paketus iš FTP kliento į serverį. Tokiu būdu buvo bandoma surasti maksimalią filtravimo reikšmę, prie kurios TCP sesija vis dar veikia stabiliai. Tyrimo metu buvo atliekamas 400 MB laikmenos persiuntimas iš FTP serverio į klientą ir palaipsniui, kas 10 % buvo mažinamas grįžtantis ACK srautas. Norint išvengti linijos persipildymo ir paketų praradimų maksimali TCP duomenų išsiuntimo sparta iš FTP serverio buvo ribojama iki 90 Mbit/s.

Siekiant, kad testavimo aplinka būtų kuo artimesnė realiam tinklui, tarp FTP serverio ir kliento buvo sudarytas mažas duomenų perdavimo tinklas iš Cisco 881 maršruto parinktuvo ir Alcatel Omnistack komutatorių.

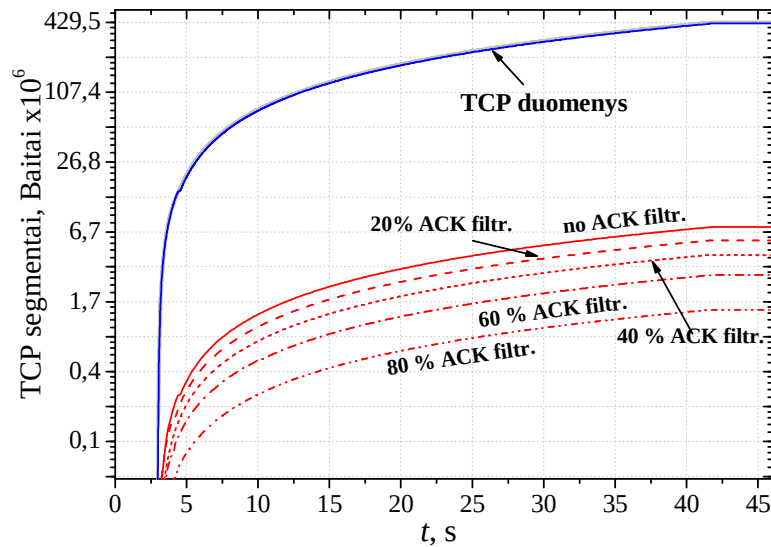
6.1 lentelė. TCP ACK paketų skaičiaus priklausomybė nuo ACK filtravimo vertės

TCP sparta, Mbit/s	FTP klientas, pps	FTP serveris, pps	Grįžtantys ACK, %
90	7456	2985	100
90	7455	2585	90
90	7456	2294	80
90	7456	2008	70
90	7455	1721	60
90	7447	1433	50
90	7461	1146	40
90	7460	860	30
90	7426	574	20
90	7426	287	10

Kaip matyti iš pateiktų tyrimo duomenų (6.1 lentelė), naudojant nustatytų sąlygų ACK filtravimą TCP sesijos grįžtančių paketų skaičius gali būti ženkliai sumažintas (iki 90 %) nedarant įtakos TCP sesijos stabilumo bei duomenų perdavimo (siunčiamų paketų) spartai. Mažinant grįžtančių ACK paketų skaičių buvo sumažintas ir duomenų srautas, sutaupant 1,4 Mbit/s tinklo pralaidos.

Didinant ACK filtravimą daugiau nei 90 %, buvo pastebėtas TCP sesijos nestabilumas ir ryškus spartos sumažėjimas. Atlikus gautų duomenų analizę buvo nustatyta, kad TCP sesijos nestabilumą ir spartos sumažėjimą sukėlė viršytas RTO laikas duomenų siuntėjo pusėje, po kurio TCP sesija pereina į pradinę – SlowStart fazę.

RTO nurodo per kiek laiko TCP duomenų siuntėjas turi gauti patvirtinimo ACK paketą. Jei per šį laiką nėra gaunamas ACK paketas, laikoma, kad išsiųstas duomenų paketas yra prarastas, o duomenų perdavimo linija yra persipildžiusi. Tokiu atveju TCP sesijos greitis sparčiai nukrenta, o duomenų siuntėjas pereina į pradinę – lėto didėjimo (angl. *Slow Start* - SlowStart) fazę. Bandant didinti TCP duomenų perdavimo spartą dėl didelio ACK filtravimo įvyksta pakartotiniai RTO laiko viršijimai. Norint nustatyti ar ryškus TCP ACK paketų sumažėjimas nesukėlė duomenų praradimų ir TCP paketų persiuntimo, buvo atliktas TCP segmentų sulginimas (16 pav.). Iš gautų duomenų matyti, kad visi tyrimo atlikti matavimai yra identiški ir ACK filtravimas nesukelia TCP sesijos paketų praradimų ir duomenų persiuntimo.



16 pav. TCP segmentų grafikas

Norint nustatyti ar ACK filtravimas yra susietas su TCP sesijos sparta, tai yra, ar maksimali ACK filtravimo procentinė reikšmė yra vienoda prie skirtingų TCP spartų buvo atlikti papildomi matavimai su skirtingomis TCP spartomis. Tam tikslui ant FTP serverio tinklo sąsajos, duomenų išsiuntimo sparta buvo mažinama palaipsniui: 90 Mbit/s, 65 Mbit/s, 32 Mbit/s, 16 Mbit/s, 8 Mbit/s 4 Mbit/s ir 1 Mbit/s. Tai buvo atliekama pasinaudojant *tc* programa.

```
tc qdisc change dev eth0 root tbf rate 90mbit burst 100kb latency 10ms \
minburst 1540
```

Su kiekviena sukonfigūruota duomenų perdavimo sparta buvo bandoma iš naujo rasti maksimalią stabilią ACK filtravimo ribą, ties kuria TCP sesija vis dar išlieka stabili. Tyrimo metu ACK filtravimo procentinė reikšmė buvo didinama žingsniškai, gauti duomenys pateikti 6.2 lentelėje.

6.2 lentelė. TCP ACK filtravimas su Cisco 881 maršruto parinktuvas

	Nenaudojant ACK filtravimo		Naudojant ACK filtravimą		
TCP sparta, Mbit/s	FTP klientas, pps	FTP serveris, pps	FTP klientas, pps	FTP serveris, pps	Grįžtantys ACK, %
90	7462	2958	7449	286	10
65	5366	2395	5366	209	9
32	2632	1317	2633	104	8
16	1316	659	1316	52	8
8	658	329	658	26	8
4	329	164	329	13	8
1	83	41	82	3	7

Kaip matyti iš gautų duomenų, ACK filtravimo lygis mažai skiriasi ir yra vidutiniškai 92 % prie skirtingų TCP spartų. Tyrimo metu buvo pastebėta, kad prie didesnių TCP spartų reikalingas didesnis grįžtančių ACK paketų skaičius.

Visi tyrimai buvo daromi naudojant spartos ribojimo mechanizmą (angl. *shaper* – shaper), kuris vėlindavo TCP paketus, bet jų neprarasdavo. Tai buvo daroma norint sukurti kuo pastovesnę perdavimo liniją, kuri leistų geriau iširti nustatytų sąlygų ACK filtravimo mechanizmą [16, 17]. Tačiau be duomenų spartos ribojimo mechanizmo įnešamo vėlinimo tinkle gali atsirasti papildomas vėlinimas dėl tarpinių tinklo įrenginių. Todėl naudojant skirtingą tinklo konfigūraciją ir įrangą, rezultatai gali skirtis. Norint tai įvertinti, tie patys matavimai buvo atlikti su Cisco 1841 ir Zyxel P2612HW maršruto parinktuvais.

6.3 lentelė. TCP ACK filtravimas su Cisco 881 maršruto parinktuvu

	Nenaudojant ACK filtravimo		Naudojant ACK filtravimą		
TCP sparta, Mbit/s	FTP klientas, pps	FTP serveris, pps	FTP klientas, pps	FTP serveris, pps	Grįžtantys ACK, %
90	7456	2993	7454	286	10
65	5366	2285	5366	209	9
32	2632	1317	2632	104	8
16	1317	578	1316	53	9
8	658	329	659	26	8
4	329	164	329	13	8
1	83	41	83	3	7

6.4 lentelė. TCP ACK filtravimas su Zyxel P2612HW maršruto parinktuvu

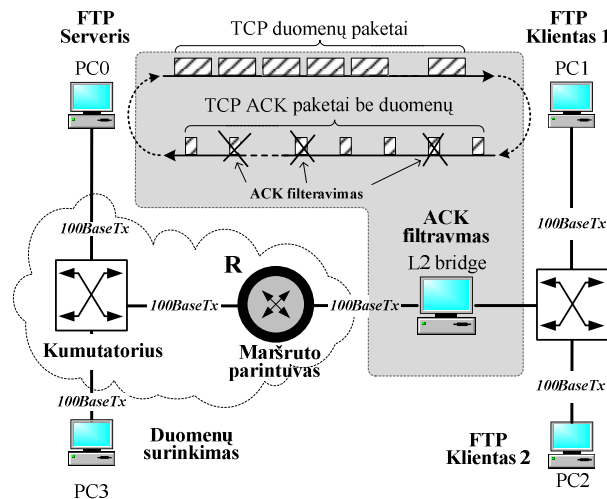
	Nenaudojant ACK filtravimo		Naudojant ACK filtravimą		
TCP sparta, Mbit/s	FTP klientas, pps	FTP serveris, pps	FTP klientas, pps	FTP serveris, pps	Grįžtantys ACK, %
90	7456	2971	7456	286	10
65	5366	2377	5366	234	10
32	2633	1315	2632	117	9
16	1316	658	1316	53	8
8	658	329	658	26	8
4	329	164	329	13	8
1	83	41	83	3	7

Kaip matyti iš gautų rezultatų skirtingi maršruto parinktuvai įneša skirtingą vėlinimą, tačiau maksimali ACK filtravimo reikšmė kinta nežymiai.

6.3. ACK filtravimas su keliomis TCP sesijomis

Šiuolaikiniai kompiuteriai ir tinklo serveriai vienu metu dažniausiai turi ne vieną, o keliolika ar kelis šimtus aktyvių TCP sesijų. Kiekviena TCP sesija yra autonominė ir konkuruoja dėl tinklo išteklių. Todėl kito tyrimo metu buvo tiriama, kaip nustatytų sąlygų ACK filtravimo mechanizmas daro įtaką kelių TCP sesijų veikimui. Tam tikslui, kaip ir pirmojo tyrimo metu buvo naudojamas Cisco

881 maršruto parinktuvas tik papildomai buvo įjungtas antras kompiuteris ir su FTP klientu. Papildomai buvo panaudotas komutatorius norint sujungti du FTP klientus į ACK filtravimo įrenginį. Kadangi komutatoriaus vėlinimas yra mažas ir jis beveik nepriklauso nuo apkrovos tai manome, kad jis neįneša papildomo vėlinimo ir nepakeičia tyrimo rezultatų.



17 pav. Tinklo schema su dvejetainiais FTP klientais

Tyrimo metu buvo atliekamas duomenų perdavimas iš serverio į klientą, tik papildomai po laiko T_1 yra aktyvuojama antra FTP sesija. Nustatytų sąlygų ACK filtravimas atliekamas atskirai kiekvienai sesijai.

Tyrimo metu buvo atliekamas pakopinis ACK filtravimo didinimas, norint nustatyti maksimalią stabilią reikšmę, prie kurios filtravimas nesukelia TCP sesijos nestabilumo ar ryškaus spartos sumažėjimo. Gauti duomenys pateikti 6.5 lentelėje.

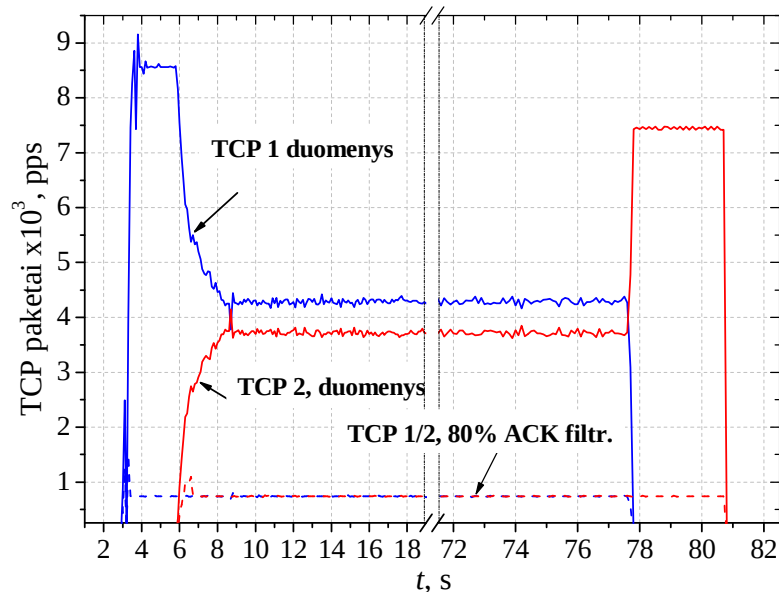
6.5 lentelė. Dviejų TCP ACK sesijų filtravimas su Cisco 881 maršruto parinktuvu

	Nenaudojant ACK filtravimo		Naudojant ACK filtravimą			
TCP sparta, Mbit/s	FTP klientas, pps	FTP serveris, pps	Pirmo FTP kliento ACK, pps	Antro FTP kliento ACK, pps	Bendras ACK srautas, pps	Grįžtantys ACK, %
90	8018	3533	---	---	---	100
90	8014	2500	566	566	1132	80
90	7455	1480	284	284	566	40
90	7455	740	142	142	284	20
90	7455	740	120	120	240	15
90	8016	370	71	71	142	10

Iš lentelėje pateiktų duomenų matyti ryškus išsiunčiamų ir gaunamų paketų skaičiaus padidėjimas, ko nebuvo pirmo tyrimo metu su viena TCP sesija. Toks ACK ir TCP duomenų paketų skaičiaus padidėjimas yra paaiškinamas tuo, kad dvi TCP sesijos konkuruoja dėl perdavimo linijos ir sukelia duomenų praradimus. Tai padidina duomenų persiuntimą ir paketų skaičių tinkle. Nors ant FTP serverio buvo naudojamas spartos ribojimo mechanizmas, tačiau jis nesugeba visiškai panaikinti paketų praradimų, nes dalis jų yra prarandama maršruto parinktuve.

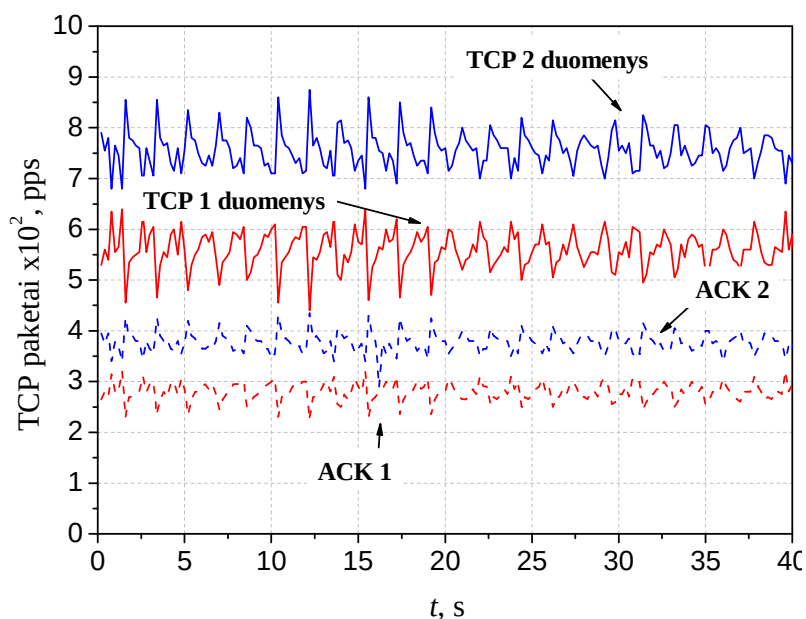
TCP ACK skaičius padidėjimą lemia ir sumažėjusi apkrova tenkanti FTP klientams. Sumažėjus apkrovai FTP klientas sugeba per tą patį laiką apdoroti gautus duomenis ir išsiųsti ACK paketą. Tai gerai matyti iš pirmo tyrimo metu gautų duomenų (6.2 lentelė), kur su mažesnėmis perdavimo spartomis TCP ACK paketų skaičius yra apytiksliai du kartus mažesnis už TCP duomenų paketų skaičių.

Iš gautų duomenų buvo nubraižytas duomenų perdavimo spartos grafikas (18 pav.).



18 pav. ACK filtravimas su keliomis TCP sesijomis

Kaip matyti iš gauto grafiko, dvi konkuruojančios TCP sesijos dažniai ne tik sukuria pastovius paketų praradimus, bet ir duomenų perdavimo linija yra dalinama netolygiai. Tai dažnas reiškinys pastebimas nefiltruotose TCP sesijose (19 pav.). To priežastis gali būti įvairūs skirtumai kompiuterių konfigūracijoje ir aparatūrinėje dalyje, todėl atsiranda nedidelis vėlinimo skirtumas, kuris ir sukuria netolygų linijos pasidalinimą. Reikėtų neužmiršti, kad viena iš TCP paskirčių yra maksimalus tinklo spartos išnaudojimas, o ne tolygus jų pasidalinimas [3, 7].



19 pav. TCP sesijos konkuravimas

Ijungus nustatytų parametrų ACK filtravimą abiem TCP sesijom, buvo pastebėtas TCP duomenų paketų skaičiaus sumažėjimas. Padidinus ACK filtravimą iki 60 % ir tik 40 % TCP ACK paketų pasiekus FTP serverį, bendras TCP duomenų paketų skaičius sumažėjo ir susilygino su pirmo tyrimo metu gautais duomenimis. Toks ryškus paketų skaičiaus sumažėjimas aiškinamas tuo, kad ACK filtravimas sumažina ACK srautą, todėl pailgėja TCP sesijos atsistatymas po paketų praradimų ir atitolinamas naujas linijos perpildymas. Maksimalus, stabilus ACK filtravimo lygis buvo pasiektas prie 80 %, kai tik 20 % bendro ACK srauto pasiekdavo FTP serverį ir kiekvienai TCP sesijai tenka tik 10 % ACK srauto. Didinant ACK filtravimą daugiau buvo pastebimas TCP sesijų nestabilumas, nors kai kuriais matavimais buvo pasiektas 5 % ACK filtravimo lygis (kai abi TCP sesijos pradedamos vienu metu).

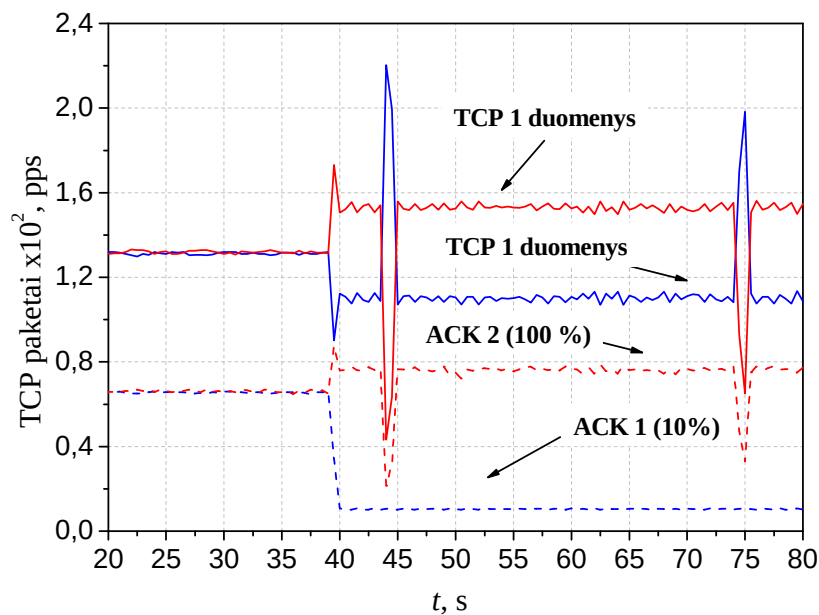
6.4. ACK filtravimo įtaka vienai TCP sesijai

Nors ACK filtravimas stabiliai ir gerai veikia su viena ar keliomis TCP sesijomis, tačiau nieko nėra žinoma kaip ACK filtravimas turi įtakos dviem TCP sesijom, iš kurių tik viena yra filtruojama. Sumažindami ACK paketų skaičių mes tiesiogiai sumažiname TCP sesijos konkurencingumą, lyginant su nefiltruota TCP sesija. Mažesnis ACK skaičius padidina atsistatymo laiką po paketų praradimų ir pailgina pradinę lėto didėjimo fazę.

Norint sužinoti ar minimali pradinė ACK sparta garantuoja pakankamą TCP sesijos konkurencingumą ir ar to užtenka, kad TCP sesija veiktų stabiliai su nefiltruojama TCP sesija buvo atlikti matavimai.

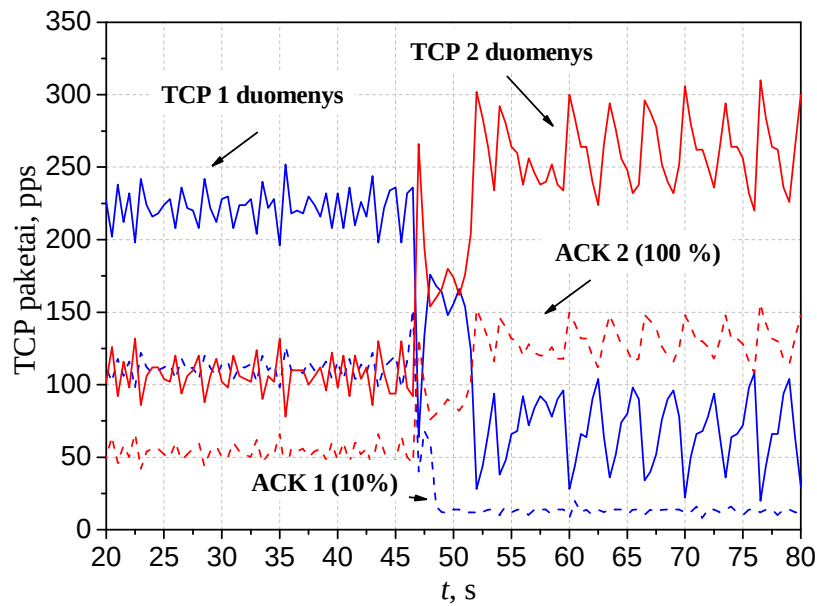
Pasinaudojus esama tinklo schema (17 pav.), buvo atliktas tyrimas, kai ACK filtravimas buvo vykdomas tik vienai TCP sesijai. Maksimalus ACK filtravimo lygis buvo pasirinktas ties 90 % riba (kuri laikoma stabili su viena ir dvejomis filtruojamomis TCP sesijomis).

Kaip matyti iš duomenų perdavimo grafiko (20 pav.), vienos TCP sesijos filtravimas pastebimai sumažino sesijos duomenų perdavimo spartą. Naudojant maksimalų – 90 % filtravimo parametą, yra stebimas 30 % spartos skirtumas tarp filtruotos ir įprastos TCP sesijos.



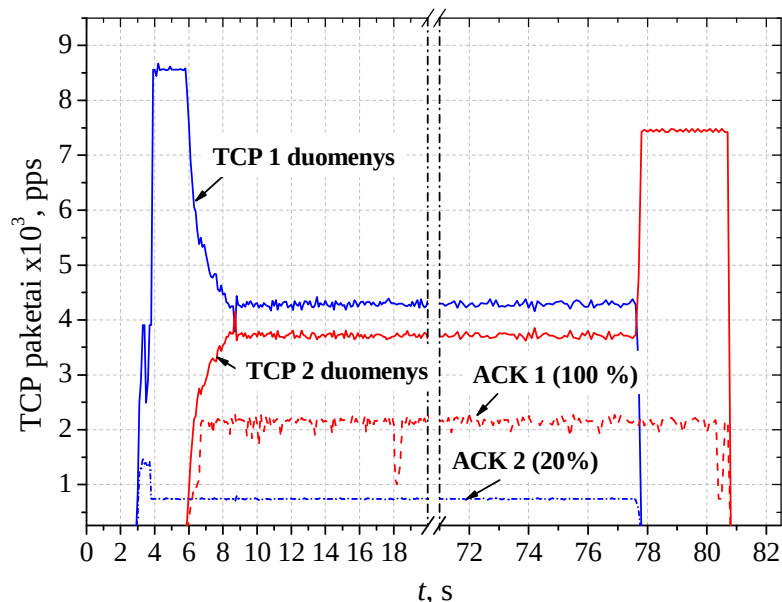
20 pav. ACK filtravimo įtaka vienai TCP sesijai

Šią išvadą patvirtina ir kiti tyrimo metu gauti rezultatai, kai naudojant maksimalų 90 % ACK filtravimą, TCP sesijos greitis ryškiai sumažėdavo. Net ir tais atvejais kai prieš įjungiant ACK filtravimą, TCP sesija būdavo konkuruojanti (21 pav.). Tai tik patvirtina iškeltą hipotezę, kad ryškiai sumažinus ACK srautus, mes sumažiname TCP sesijos konkurenciją.



21 pav. ACK filtravimo įtaka vienai TCP sesijai

Norint sušvelninti ACK filtravimo įtaką, ir padidinti TCP sesijos konkurencingumą, ACK filtravimo lygis buvo sumažintas iki 80 %, praleidžiant 20 % ACK į TCP serverį (22 pav.). Toks sprendimas padidino TCP sesijos atstatymo greitį ir konkurencingumą lyginant su nefiltruojama TCP sesija. Kaip matyti 22 pav., greičių skirtumas tarp dviejų TCP sesijų yra pakankamai mažas ir siekia apytiksliai 13 %.



22 pav. ACK filtravimo įtaka vienai TCP sesijai

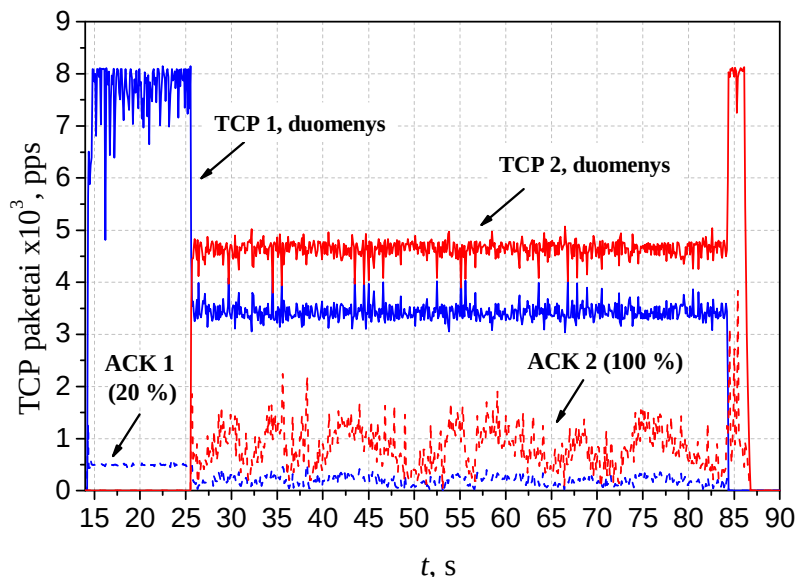
Didinant praleidžiamų ACK paketų skaičių, spartų skirtumas tarp TCP sesijų gali būti dar daugiau sumažintas, taip padidinant filtruojamos TCP sesijos konkurencingumą.

6.5. TCP ACK filtravimas nehomogeninėse aplinkose

Nors nustatytų sąlygų ACK filtravimas korektiškai veikia ir sudėtinguose sąlygose su dvejomis TCP sesijomis iš kurių tik viena yra filtruojama, tačiau visi atlikti tyrimai buvo atliekami idealioje aplinkoje, kur paketų praradimai ir vėlinimas buvo minimalūs ir kontroliuojami.

Dažnas duomenų perdavimo tinklas yra nehomogeninis, su pastoviais besikeičiančiais parametrais. Tokiuose tinkluose dažnai TCP duomenų perdavimo sparta yra ribojama ne spartos ribojimo mechanizmu, bet panaudojant agresyvesnį nustatytų parametrų ribotuvą (angl. *policer* – policer). Toks TCP duomenų srauto greičio ribojimas sukelia didelius paketų praradimus ir padidina duomenų persiuntimą. Esant mažam vėlinimui, TCP protokolas sugeba sėkmingai prisitaikyti prie tokių aplinkybių ir pakankamai optimaliai išnaudoti tinklo perdavimo spartą.

Norint sužinoti ar ACK filtravimo mechanizmas gali sėkmingai veikti nehomogeniniuose tinkluose ir optimaliai išnaudoti tinklo išteklius, buvo atliktas tyrimas su dviem TCP sesijomis, iš kurių tik vieną buvo filtruojama. Norint sudaryti duomenų linijos persipildymus ir paketų praradimus ant FTP serverio tinklo sąsajos buvo išjungtas idealus spartos ribojimo mechanizmas. Tai sukėlė papildomus tinklo persipildymus ir paketų praradimus. ACK filtravimo riba buvo parinkta ties 80 %, nes iš praeitų tyrimų parodė gerus rezultatus.



23 pav. TCP ACK filtravimas nehomogeniniuose terpėse

Atliekant matavimus, pirma TCP sesija buvo sudaryta iš FTP kliento, kurio grįžantis ACK srautas buvo filtruojamas ties 80 % riba. Po trumpo laiko T_1 tarpo buvo sudaryta antra TCP sesija iš antrojo kompiuterio, kuris nebuvo filtruojamas. Duomenų perdavimo grafike (23 pav.) matyti, kad abi TCP sesijos buvo nepastovios ir pastoviai konkuravo dėl perdavimo linijos. Dažni sparšai

pakitimai („dryžiai“) matomi grafike nurodo, kad duomenų perdavimo metu buvo daug paketų praradimų ir vyko pakartotinis persiuntimas.

Iš gautų duomenų galime teigti, kad TCP sesija išlieka stabili ir konkurencinga nehomogeniniuose aplinkose, kai TCP ACK filtravimo lygis neviršija 80 %. Didesnis ACK filtravimo dydis galimas tik pavieniauose TCP sesijose, kai perdavimo linija yra be paketų praradimų ar filtruojama tik viena TCP sesija.

6.6. ACK filtravimo įtaka trumpų sesijų spartai

Dauguma atliktų tyrimų [9–11] analizuoja kaip ACK filtravimo mechanizmas daro įtaką TCP sesijos stabilumui, kai duomenų siuntėjo langas (CWND) yra visiškai atsidaręs ir TCP sesija yra persipildymo vengimo fazėje. Tačiau didžioji dalis TCP sesijų internete yra labai trumpos ir neužtrunka daugiau nei sekundę. Tokios TCP sesijos visą duomenų siuntimo laiką būna pradinėje fazėje, kai yra didinama duomenų išsiuntimo sparta. TCP sesijos augimo greitis SlowStart fazėje tiesiogiai priklauso nuo grįžtančio ACK paketų skaičiaus ir vėlinimo. Filtruodami grįžtančius TCP ACK paketus mes tiesiogiai mažiname TCP spartą ir pailginame sesijos laiką.

Norint nustatyti kaip ir kiek nustatytų sąlygų ACK filtravimas daro įtaką trumpoms TCP sesijoms, buvo atlikti mažų informacijos laikmenų perdavimai laike matavimai kiekvieną kartą matuojant pirmo ir paskutinio TCP paketo priėmimo laikus. Pasinaudojus pirmo tyrimo sukurta testine aplinka buvo atlikti matavimai. Maksimali duomenų išsiuntimo sparta buvo ribojama iki 90 Mbit/s. Nustatytų sąlygų ACK filtras buvo sukonfigūruotas praleisti iki 20 % grįžtančių ACK arba apytiksliai 570 paketų per sekundę. Nors maksimali, nefiltruojama 570 ACK paketų per sekundę sparta garantuoja apytiksliai 12 Mbit/s spartą SlowStart fazėje, kai ACK paketas yra siunčiamas kiekvienam antram TCP segmentui. Trumpose sesijose, kur duomenų persiuntimas yra milisekundžių eilės, ši vertė gali būti daug mažesnė ir nesiekti 1 Mbit/s.

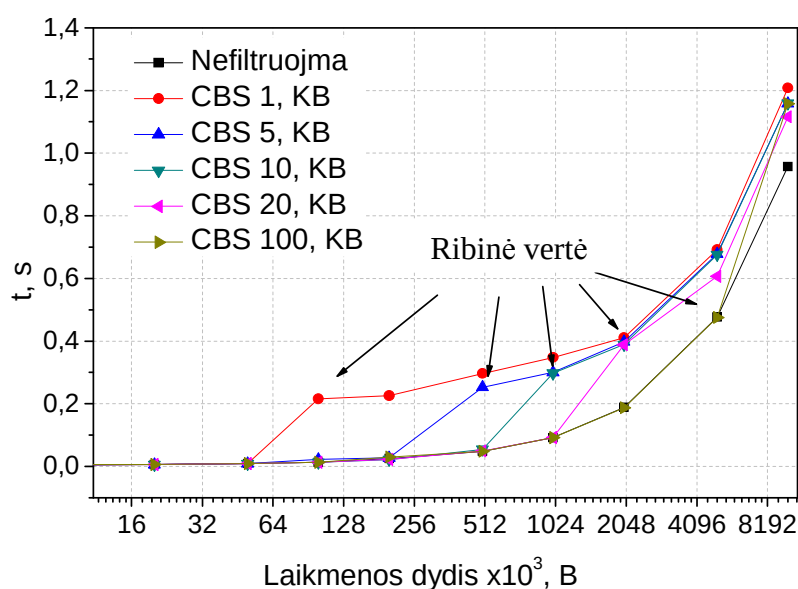
Kadangi TCP spartos augimas pradinėje fazėje tiesiogiai priklauso nuo gaunamų paketų skaičiaus tai tyrimo metu buvo keičiamas CBS parametras, kuris nurodo nustatytų sąlygų ACK filtro buferio dydį baitais.

6.6 Duomenų siuntimo laiko priklausomybė nuo CBS parametro

	Laikas, s					
Laikmenos dydis $\times 10^3$, Baitais	Nefiltruojama	CBS 1, KB	CBS 5, KB	CBS 10, KB	CBS 20, KB	CBS 100, KB
1	0,0037	0,0047	0,0035	0,0047	0,0037	0,0037
2	0,0037	0,0037	0,0037	0,0037	0,0037	0,0037
5	0,004	0,004	0,004	0,004	0,004	0,004
10	0,005	0,005	0,005	0,0057	0,0057	0,0057
20	0,0062	0,0062	0,0062	0,0062	0,0062	0,0062
50	0,009	0,009	0,009	0,0087	0,0087	0,0087
100	0,0132	0,2151	0,0219	0,0132	0,0132	0,013
200	0,0247	0,2258	0,0274	0,0217	0,0219	0,03
500	0,0482	0,2968	0,2526	0,0542	0,0499	0,0477
1000	0,0919	0,3475	0,3	0,2971	0,0917	0,0919
2000	0,1879	0,411	0,3977	0,389	0,3883	0,1876
5000	0,4762	0,6911	0,6778	0,676	0,6062	0,4759
10000	0,9572	1,208	1,1583	1,1586	1,1157	1,157

Kaip matyti šio tyrimo metu gautų duomenų, egzistuoja ribinė vertė, nuo kurios TCP sparta arba užtrukęs laikas pastebimai padidėja. Ši „ribinė“ vertė priklauso nuo CBS parametro dydžio ir keičiant ją mes galime pastumti „ribinę“ vertę į mažesnių ar didesnių laikmenų pusę.

Visais atliktais ACK filtravimo atvejais ryškus siuntimo laiko padidėjimas buvo sukeltas dėl RTO laiko viršijimo. Tai sukėlė apytiksliai 0,2 s siuntimo laiko padidėjimą. Didindami CBS parametą, mes didiname RTT ir RTO laikus, todėl atitoliname duomenų persiuntimą.



24 pav. Duomenų laikmenos persiuntimų laikas

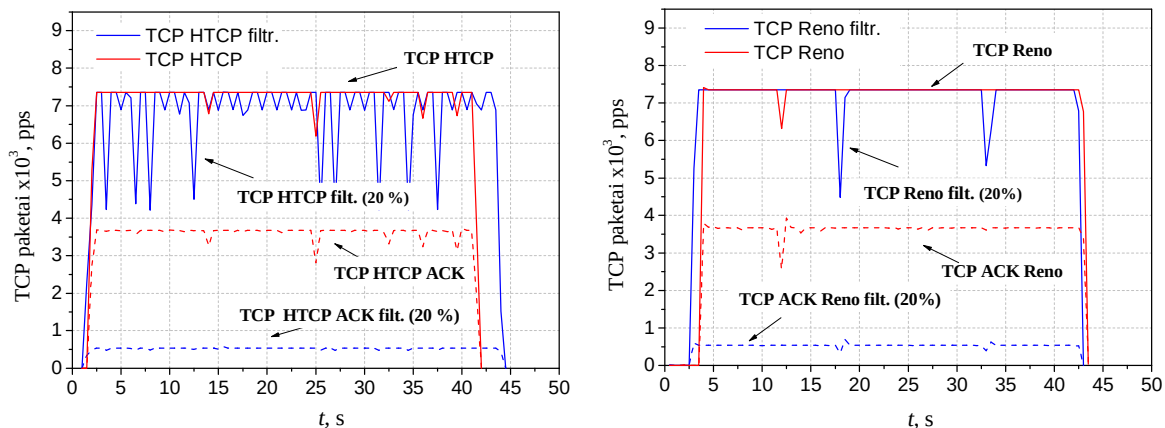
Iš tyrimo metu gautų duomenų matyti, kad ACK filtravimas sukelia TCP sesijos spartos sumažėjimą, kas yra nepageidaujama trumpų sesijų metu. Tačiau keičiant CBS parametą mes galime šią „ribinę“ vertę pastumti į didesnių ar mažesnių laikmenų pusę. Tačiau didinant CBS vertę reikėtų neužmiršti, kad tai padidina ir vėlinimą, kuris TCP sesijai yra nepageidaujamas parametras kuris turi neigiamai paveikti TCP spartai.

6.7. ACK filtravimas su skirtingais persipildymo vengimo algoritmais

Darbo metu visi atlikti tyrimai su TCP ir ACK filtravimu, buvo atliekami su Linux FTP serveriu, kuris veikia su CUBIC persikildymo vengimo algoritmu. TCP CUBIC algoritmas yra diegiamas į visus Linux OS, kaip tipinis persipildymo vengimo algoritmas. Tačiau be TCP CUBIC egzistuoja daug kitų persipildymo vengimo algoritmų, kurie nuolat tobulėja ir yra keičiami pagal tinklo ir sistemos poreikius [15].

Norint sužinoti ar ACK filtravimas daro įtaką kitiems TCP persipildymo vengimo algoritmų veikimui – TCP spartą, buvo atlikti praktiniai matavimai. Pasinaudojant esama tinklo schema, buvo atliekamas 400 MB laikmenos atsisiuntimas iš serverio. Norint kad matavimo rezultatai būtų tikroviškesni, duomenų perdavimo linija buvo apribota iki 90 Mbit/s su mažu paketų praradimu, o abu tinklo elementai (FTP klientas ir serveris) veikia tuo pačiu persipildymo vengimo algoritmu.

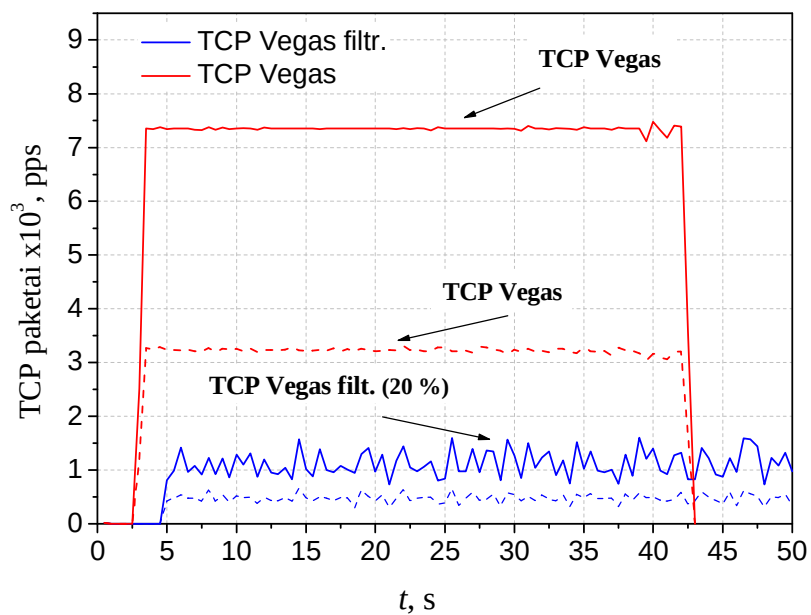
Tyrimams buvo pasirinkti skirtingi persipildymo vengimo algoritmai: *Vegas*, *Reno*, *HTCP*, ir atlikti du matavimai, su įjungtu ir išjungtu 80 % ACK filtru.



25 pav. TCP HTCP (kairė) ir Reno (dešinė) algoritmai

Iš gautų rezultatų buvo nubraižyti grafikai (25 pav.) kuriuose atvaizduoti TCP duomenų perdavimo sparta. Nors iš gautų duomenų matomas nedidelis paketų skaičiaus praradimas, tačiau didesnių spartos sumažėjimų nebuvo pastebėta.

Atliekant identiška tyrimai su TCP *Vegas* algoritmu (26 pav.), buvo pastebėtas ryškus spartos sumažėjimas. Tokią spartos nukritimą lemia algoritmo veikimo mechanizmas, kuris veikia tiesiogiai nuo grįžtančių ACK paketų skaičiaus.



26 pav. TCP Vegas algoritmas

Iš tyrimo gautų duomenų matyti, kad ACK filtravimas gali veikti netik su TCP CUBIC, bet ir su kitais TCP persipildymo vengimo algoritmais. Nors su TCP Vegas algoritmu buvo matomas didelis spartos sumažėjimas, tačiau reikėtų pastebėti, kad šis algoritmas yra skirtas veikti pagal grįžtančius ACK paketus [19, 20], o ne RTT laiko ar sėkmingai persiūtų segmentų skaičiaus matavimus.

7. ACK FILTRAVIMO ĮTAKA MARŠRUTO PARINKTUVO APKROVAI

Kaip žinoma tinklo maršruto parinktuvo apkrova tiesiogiai priklauso nuo perduodamų paketų skaičiaus (pps), bet dažniausiai ne nuo jų dydžio [21–23]. Nagrinėjant ACK filtravimą buvo nustatyta, kad stabilus ACK filtravimas galimas prie 80 %. Ir toks ryškus ACK paketų sumažėjimas tinkle dažniai neturi šalutinių efektų ir gali stabiliai veikti ir su nefiltruojamomis TCP sesijomis.

Šiuo tyrimo metu bus bandoma rasti priklausomybę tarp ACK filtravimo ir maršruto parinktuvo CPU apkrovos, kaip ir kiek tinklo išteklių mes galime sutaupyti naudojant nustatytą sąlygų ACK filtravimą.

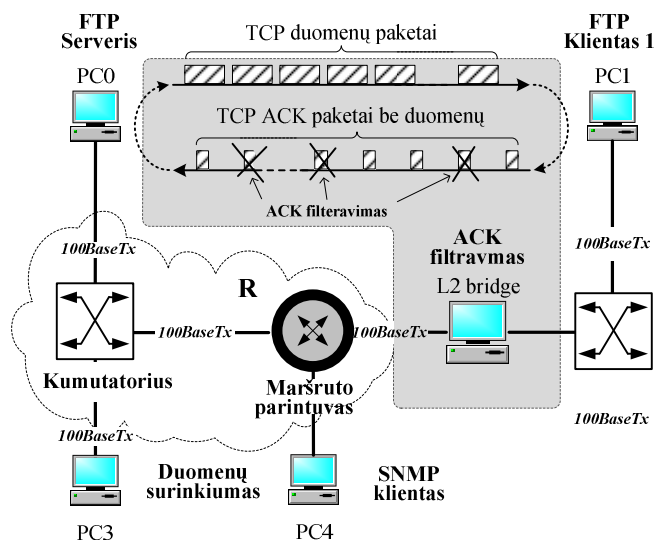
7.1. ACK filtravimo įtaka maršruto parinktuvo apkrovai

Praktikoje nagrinėjant maršruto parinktuvo apkrovos priklausomybes nuo paketų skaičiaus, dažniausia yra remiamasi RFC 2544 dokumentu. Tokio tyrimo metu yra generuojamas UPD srautas ir atliekamas apkrovos matavimas.

Mūsų tyrimo metu yra nagrinėjamas TCP protokolas, kuris priešingai nei UDP yra dvikryptis ir asimetrinis. TCP protokolo sparta tiesiogiai priklauso ir nuo kitų tinklo parametrų tokių kaip vėlinimas, vėlinimo deviacija bei paketų praradimai.

Norint nustatyti maršruto parinktuvo apkrovos priklausomybę nuo TCP duomenų perdavimo spartos buvo atlikti tyrimai su skirtingais įrenginiais prie skirtingų perdavimo spartų [25, 26].

Tyrimo metu buvo pasirinkti trys skirtingi tinklo maršruto parinktuvai: Cisco 881, Cisco 1841 ir Zyxel P2612HW. Kiekvienas iš jų turėjo identišką konfigūraciją, o visi sisteminiai apkrovos matavimai buvo atlikti SMTP protokolu. Spartos ir apkrovos matavimai buvo atlikti ant esamo tinklo schemos (27 pav.), tik papildomai buvo matuojama maršruto parinktuvo apkrova SNMP protokolu.



27 pav. Tinklo testavimo schema

Nors maršruto parinktuvo apkrova taip pat priklauso ir nuo aktyvių TCP sesijų skaičiaus, tačiau tyrimas buvo atliekamas su viena aktyvia TCP sesija, norint išlaikyti nesudėtingą, kontroliuojamą testavimo aplinką be paketų praradimų.

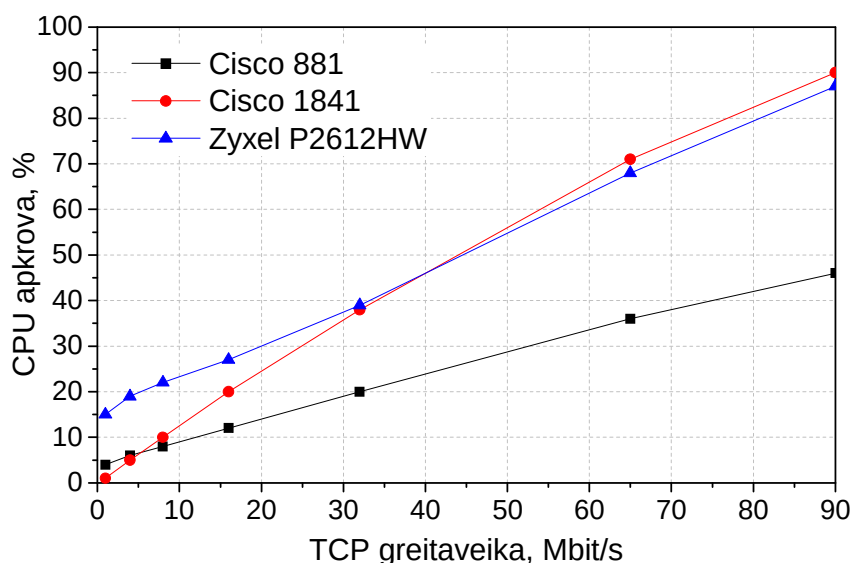
Pirmo tyrimo metu buvo atliktas pakopinis TCP spartos mažinimas, matuojant kaip TCP sparta daro įtaką įrenginių apkrovai. Gauti duomenys pateikti 7.1 lentelėje.

7.1 lentelė. Maršruto parinktuvo apkrovos priklausomybė nuo TCP spartos

	Standartinis veikimas				
Sparta, Mbit/s	FTP klientas, pps	FTP serveris, pps	Cisco 881 CPU %	Cisco 1841 CPU %	Zyxel P2612HW CPU %
90	7462	2958	46	90	87
65	5366	2395	36	71	68
32	2632	1317	20	38	39
16	1316	659	12	20	27
8	658	329	8	10	22
4	329	164	6	5	19
1	83	41	4	1	15

Iš tyrimo metu gautų rezultatų matyti, kad visų maršrutų parinktuvų apkrova yra beveik tiesiška ir priklauso nuo per juos persiunčiamų duomenų – paketų skaičiaus. Prie mažesnių TCP spartų (28 pav.)

pastebimas nedidelis netiesiškumas, kuris gali atsirasti dėl vidinių maršruto parinktuvo sisteminių procesų (SNMP protokolų užklauskos, vidinės sisteminės pertrauktys).



28 pav. Maršruto parinktuvo CPU apkrova nuo TCP spartos

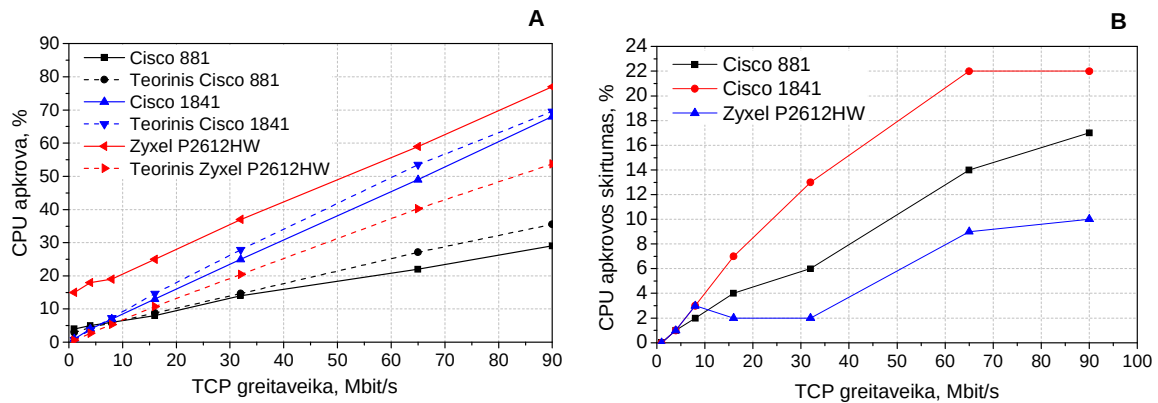
Žinant, jog skirtingos spartos generuoja skirtingas apkrovas, buvo atlikti pakartotiniai tyrimai su įjungtu 80 % ACK filtravimo mechanizmu ir išmatuotos apkrovos. Iš duomenų pateiktų 7.2 lentelėje matyti ryškus sumažėjimas visuose tirtuose įrenginiuose, kaip ir buvo tikimasi.

7.2 lentelė. Maršruto parinktuvo apkrova naudojant ACK filtravimą

TCP sparta, Mbit/s	Naudojant ACK filtravimą				
	FTP klientas, pps	FTP serveris, pps	Cisco 881 CPU, %	Cisco 1841 CPU, %	Zyxel P2612HW CPU, %
90	7462	2958	29	68	77
65	5366	2395	22	49	59
32	2632	1317	14	25	37
16	1316	659	8	13	25
8	658	329	6	7	19
4	329	164	5	4	18
1	83	41	4	1	15

Nors apkrovos sumažėjimas matomas visuose trijuose įrenginiuose, tačiau paskaičiavus teorinę ir praktinę apkrovą, kurią turėtume gauti naudojant ACK filtravimą, matomas netolygumas (29

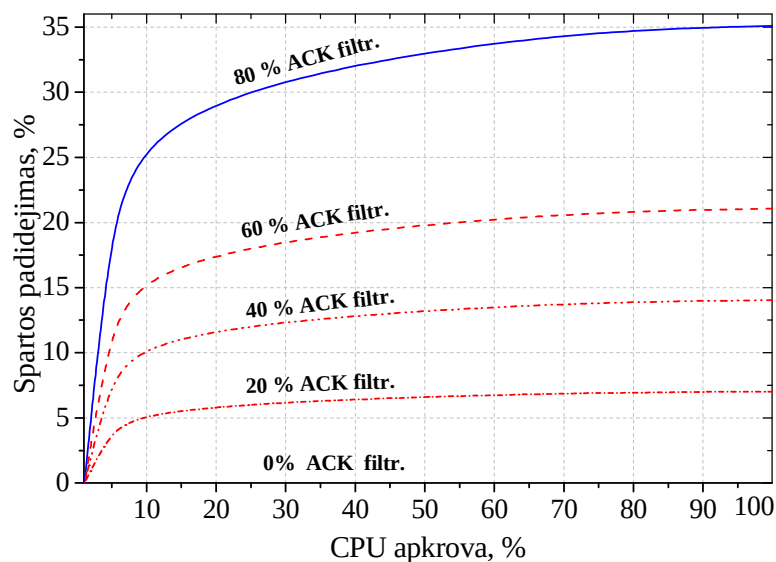
paveikslėlis A). Šis skirtumas atsiranda dėl skirtingų aparatūrinių ir programinių skirtumų skirtinguose maršrutų parinktuvuose [27, 28].



29 pav. ACK filtravimo įtaka maršruto parinktuvo apkrovai

Iš procentinės išraiškos (29 paveikslėlis B), yra gerai matoma, kad idealiausiai teorinį modelį atitinka Cisco 881 maršruto parinktuvas. Jo procentinė išraiška gali būti laikoma tiesiška, lyginant su kitais maršruto parinktuvais.

Norint geriau ištyti CPU apkrovos priklausomybę nuo ACK filtravimo lygio ir TCP spartos, buvo atlikti skirtingi matavimai su Cisco 881 maršruto parinktuvu. Tyrimo metu buvo naudojamas idealus tinklo ribotuvas, kuris keitė TCP duomenų perdavimo spartą. ACK filtravimo vertė buvo keičiama nuo 0 % iki 80 %, kiekvieną kartą matuojant maršrutizatoriaus apkrovą.



30 pav. Apkrovos priklausomybė nuo ACK filtravimo

Iš tyrimo metu gautų duomenų buvo nubraižytas apkrovos sumažėjimo priklausomybės grafikas nuo ACK filtravimo lygio (30 pav.). Jis parodo kiek sisteminių išteklių galima sutaupyti maršruto parinktuve (Cisco 881), naudojantis vieną ir kitą ACK filtravimo vertę prie skirtingų TCP perdavimo spartų.

Esant teoriniam 100 % CPU apkrovai (kas laikoma nestabilia maršruto parinktuvo darbine aplinka), naudojant maksimalų 80 % ACK filtravimą, apkrova gali būti sumažinta net 35 % (iki stabilios, darbinės apkrovos). Naudojant nustatytą sąlygų ACK filtravimą, praktiškai įmanoma sutaupyti sisteminius maršruto parinktuvo išteklius arba padidinti tinklo pralaidą net iki 35 %.

8. APIBENDRINIMAS. IŠVADOS

Magistro darbe nagrinėjamas transportinio lygmens duomenų perdavimo TCP protokolas ir jo funkcijos. Pirmoje darbo dalyje trumpai apžvelgti pagrindiniai protokolo laukai, jų reikšmės, išanalizuoti pagrindiniai mechanizmai ir algoritmai kurie naudojami sėkmingam duomenų perdavimui. Trumpai aptartos pagrindinės problemos ir trūkumai, su kuriomis susiduria TCP protokolas. Viena iš problemų yra per dažnas ACK paketų generavimas.

Antroje darbo dalyje trumpai apžvelgti TCP ACK srauto mažinimo mechanizmai ir būdai, kaip alternatyva esamiems sprendimams, buvo pasiūlytas visiškai naujas – nustatytų sąlygų ACK filtravimo mechanizmas. Be esamų ACK filtravimo panaudojimo galimybių pasiūlytos naujos šio mechanizmo pritaikymo galimybės praktikoje. Kai ACK filtravimas yra naudojamas simetrinėse, didelės spartos perdavimo linijose, norint sumažinti tarpinių ir galinių tinklo įrenginių (ugniasienių, maršruto parinktuvų, serverių ir t.t.) apkrovą. Siekiant patikrinti šias hipotezes, realiame duomenų perdavimo tinkle atlikti praktiniai tyrimai. Remiantis tyrimo metu gautais rezultatais suformuluotos šios išvados (magistro tezės):

1. TCP ACK filtravimo mechanizmas, neįtakoja TCP sesijos spartos ir stabilumo, galima sėkmingai filtruoti net iki 80 % grįžtančių ACK paketų. Didinant ACK filtravimą virš 85-90 % pastebimas TCP sesijos nestabilumas ir ryškus spartos sumažėjimas. Tai atsitinka dėl viršyto TCP RTO laiko;
2. ACK filtravimas sėkmingai veikia net tik su viena TCP sesija, bet ir su keliomis TCP sesijomis, nepriklausomai ar ACK filtravimas naudojamas visose ar tik vienoje TCP sesijų. Pažymėtina, kad filtruojama(-os) TCP sesija išlieka konkurencinga kitų sesijų atžvilgiu;
3. Didžiausia įtaka TCP duomenų perdavimo spartai naudojant nustatytų sąlygų ACK filtravimą yra pastebima trumpų TCP sesijų metu, kai duomenų siuntimo langas yra nepakankamai „atsidaręs“. TCP sesijos siuntimo laikas tiesiogiai priklauso nuo CBS parametro ir persiunčiamos laikmenos, duomenų kiekio. Keičiant CBS vertę, ribinę paketo dydžio vertę gali būti pakeista į mažesnių ar didesnių laikmenų pusę;
4. ACK filtravimo mechanizmas sėkmingai veikti netik su TCP CUBIC, bet ir kitais persipildymo vengimo (*Vegas*, *HTCP*) algoritmais;
5. Maršruto parinktuvo apkrova yra tiesinė ir priklauso nuo paketų skaičiaus per sekundę. Didėjant TCP perdavimo spartai, vis didesni duomenų perteklių generuoja TCP ACK paketai, kurie sukelia nereikalingą apkrovą maršruto parinktuvui. Norint sumažinti šį duomenų perteklių ir atlaisvinti maršruto parinktuvo išteklius, ACK filtravimas gali būti naudojamas;

6. Efektyviai naudojant ACK filtravimo mechanizmą maršruto parinktuvo apkrova galima sėkmingai sumažinta iki 32 % nuo esamos maksimalios procentinės apkrovos vertės. Maršruto parinktuvo apkrovos sumažėjimas yra netiesinis ir priklauso nuo pradinės apkrovos lygio bei ACK filtravimo vertės. Veikiant maršruto parinktuvui ties 60 % CPU apkrova pritaikius maksimalų stabilų ACK filtravimo lygį (80 %), CPU apkrova gali sumažėti iki 40 %. Tačiau prie 10 % apkrovos bus gautas tik 25 % apkrovos sumažėjimas, t.y. iki 8 %.

Apibendrinant darbo rezultatus reikėtų pastebėti, kad įprastomis tinklo veikimo sąlygomis ACK filtravimo mechanizmas nedaro įtakos TCP sesijos stabilumui bei spartai ir gali žymiai padidinti tinklo bei jų įrenginių efektyvumą. O lankstus ir lengvas realizavimo mechanizmas leidžia jį lengvai pritaikyti praktikoje.

Literatūra

1. DALTON L. A.; ISEN C. A. 2004. Study on High Speed TCP Protocols, *Conference IEEE: GLOBECOM*, – Vol. 2. – P. 851-855.
2. SANGTAE H.; INJONG R.; LISONG X. 2008. CUBIC: A New TCP-Friendly High-Speed TCP Variant, *Conference SIGOPS*, – Vol. 42, No. 5. – P. 64-74.
3. ESCHEIKH M.; BARKAOUI K. 2007. Performance analysis of high-speed TCP protocols BIC and CUBIC with AQM in lossy networks, *Proceedings of the IASTED International Conference on Communication System*, P. 31-35
4. LIBOR D.; ALENA K. 2006. Understanding TCP/IP, 480 p.
5. LUIGI A.; GRIECO, S. M. 2004. Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control, ACM CCR.
6. MANISH J.; RAVI S. P.; CONSTANTINOS D. 2003. The TCP Bandwidth–Delay Product revisited: network buffering, cross traffic, and socket buffer auto–sizing, P 1-15.
7. SANGTAE H.; INJONG R.; LISONG X. 2008. CUBIC: a new TCP-friendly high-speed TCP variant, *ACM SIGOPS Operating Systems Review*, Vol. 42, P. 64-74.
8. CHEN B.; MARSIC I.; SHAO H-R.; MILLER R. 2009. Improved Delayed ACK for TCP over Multi-hop Wireless Networks, *Conference IEEE: WCNC*, P. 1-5.
9. WU' H.; WU' J.; CHENG' SH.; MA J. 1999. ACK Filtering on Bandwidth Asymmetry Networks, *Conference APCC/OECC*, P. 175-178.
10. KECALI F.; INAN I.; AYANOGLU E. 2007 TCP ACK Congestion Control and Filtering for Fairness Provision in the Uplink of IEEE 802.11 Infrastructure Basic Service Set *Conference IEEE: ICC*, P. 4512-4517.
11. KARN P. 1996, [interaktyvus] Mailing list, Subject: dropping TCP ACK's // end2end-interest, karn@qualcomm.com., [žiūrėta: 2010-05-03]. Prieiga per internetą: <ftp://ftp.isi.edu/end2end/end2end-interest-1996.mail>
12. MARTIN A. B. 2006. Traffic Control using tcng and HTB HOWTO, [interaktyvus]. [žiūrėta: 2010-05-12]. Prieiga per internetą: < <http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Traffic-Control-tcng-HTB-HOWTO.pdf>>
13. IETF RFC 2475, 1998. An Architecture for Differentiated Services, 36 p.
14. IOANNIS P.; VASSILIS T. The TCP Minimum RTO Revisited, *Networking 2007*, P. 981-991.
15. KIM, H.; CLAFFY K.; FOMENKOV M.; BARMAN, D.; FALOUTSOS, M.; AND LEE, K. 2008. Internet traffic classification demystified: myths, caveats, and the best practices *Conference CoNEXT '08: New York: ACM: ACM*, P. 1-12.
16. CACERES R.; DANZIG P. B.; JAMIN S.; MITZEL D. J. 1991. Characteristics Of Wide-Area TCP/IP Conversations, *Conference ACM SIGCOMM*, P. 101-112.

17. XIAO Y.; ROSDAHL J. 2002 Throughput and delay limits of IEEE 802.11, *IEEE: Communications Letters*, Vol.6, Is. 8, P. 355-357.
18. CHATZIMISIOS P.; VITSAS V.; BOUCOUVALAS A. C. 2002. Throughput and delay analysis of IEEE 802.11 protocol, *5th Int Conference IEEE: Networked Appliances*, P. 168- 174.
19. BENCIVENNI M.; CARBONE A.; FELLA A.; GALLI D.; MARCONI U.; PECO G., PERAZZINI S.; VAGNONI V.; ZANI S. High rate packet transmission on 10 Gbit/s Ethernet LAN using commodity hardware, *16th Conference IEEE: Real Time, IEEE-NPSS*, P. 167-182.
20. LAWRENCE S.; BRAKMO LARRY L. 1995. TCP Vegas: End to End Congestion Avoidance on a Global Internet, *PetersonIEEE Journal on Selected Areas in Communication*, Vol. 13, No 8. P 1465-1480.
21. OMAR, A. H.; EITAN A. 2004. Anglysis of TCP Vegas and Reno, *Telecommunication Systems 15*, P. 381-404.
22. PAREDES-FARRERA M.; FLEURY M. 2006. Ghanbari M. Router Response to Traffic at a Bottleneck Link, *2nd Int. Conference IEEE: TRIDENTCOM*, P. 4-41.
23. OGAWA Y.; NAKAYA A. 2002 Estimating the Performance of a Large Enterprise Network for the Updating of Routing Information, *Conference Workshop IEEE: IP Operations and Management*, P. 161-165.
24. JACOBSON V. 1988. Congestion avoidance and control, *Conference SIGCOMM' 88: ACM Special Interest Grp. on Data Comm. New York: ACM*, P. 314-329.
25. BERT H. 2002. Linux Advanced Routing & Traffic Control HOWTO, 158 p.
26. YEE-TING L.; DOUGLAS L.; ROBERT N. S. 2005. Experimental Evaluation of TCP Protocols for High-Speed Networks, HI Technical Report.
27. KRISHNAN K.; ASHOK K. A.; RAGHAVAN S. V. 1995m. A generic architecture for programmable traffic shaper for high-speed networks, *Computer Science Technical Report Series; Vol. CS-TR 3495*, 11 p.
28. JOHN W. L. 2000. Evolvable Internet Hardware Platforms, *The Third NASA/DoD Workshop on Evolvable Hardware, EH'2001*, 9 p.
29. FRED K.; JOHN D.; ANSHUL K.; RALPH K.; JOHN L.; PRASHANTH P.; DAVID R.; DAVID T.; JYOTI P.; ED S.; JON T.; KEN W. 2002. Design of a High Performance Dynamically Extensible Router, *Proceedings of the DARPA Active Networks Conference and Exposition*, 23 p.

PRIEDAI

A priedas. Pranešimo Evaluation of TCP acknowledgment mechanism influence on router performance. TELECOMMUNICATIONS ENGINEERING – T 180.



DIPLOMAS



LMA



KTU



VGTU

TALINNA TEHNIKAKOOL



TU



RTU

LIETUVOS MOKSLŲ AKADEMIJA

KAUNO TECHNOLOGIJOS UNIVERSITETAS

RYGOS TECHNIKOS UNIVERSITETAS

TALINO TECHNOLOGIJOS UNIVERSITETAS

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Nuoširdžiai dėkoja už aktyvų dalyvavimą
tarptautinėje tryliktojoje konferencijoje
"ELEKTRONIKA 2010"
ir perskaitytą mokslinį pranešimą

**Evaluation of TCP Acknowledgment
Mechanism Influence on Router Performance**

dalyviams, profesoriams

L. PAVILANSKAS, A. STATKUS

Konferencijos pirmininkas
akademikas, prof. D. Eidukas

Telekomunikacijų ir elektronikos fakulteto dekanas
prof. A. Valinevičius



KAUNAS, 2010 m. gegužės 18-20 d.



The Institute of Electrical and Electronics Engineering, Inc.
jungtinio Signalų apdorojimo – Skaitinio intelekto – Ryšių padalinio taryba
šiuo raštu iškilmingai patvirtina, kad

Arūnas STATKUS

laimėjo

*antrąją premiją
geriausio jaunojo mokslininko pranešimo konkurse*

už pranešimą „Evaluation of TCP Acknowledgment Mechanism Influence on Router Performance“
pristatytą 2010 m. gegužės 20 d. Vilniuje vykusios keturioliktosios tarptautinės konferencijos
ELEKTRONIKA 2010 sekcijoje „Telekomunikacijų inžinerija – T180“

I. Naidionova
Lietuvos sekcijos pirmininkė

D. Navakauskas
Lietuvos SP/CIS/COM padalinio pirmininkas

Evaluation of TCP acknowledgment mechanism influence on router performance

L. Pavilanskas, A. Statkus

*Telecommunications Engineering Department, Vilnius Gediminas Technical University,
Naugarduko str. 41, LT-03227 Vilnius, Lithuania, E-Mail: lukas.pavilanskas@el.vgtu.lt, arunas.statkus@teo.lt*

Introduction

Nowadays with growth of data traffic, it is essential to control channel congestion efficiently in Internet for successful utilizing network resources (routers, switches, links, and etc). The main protocol for current objective is Transport Control Protocol (TCP) today. It is reliable connection-oriented protocol, which implements the flow control on heterogeneous network by means of a sliding window, congestion avoidance, and acknowledgment mechanisms, simultaneously. TCP runs over more network technologies than any other protocol and provides the highest degree of interoperability.

The main problem with TCP is that typical protocol implementations (RFC 3782) are successful at low speed (up to 100 Mbps) but it is unfit for high speed networks due to slow grow-up of congestion window, and poor evaluation of upper bound of the channel [1]. This is because the congestion window growth function – $W(t)$ has square features on initial phase (RFC 2581) and depends on channel delay: if latency is high, the congestion window growth rate will be low. To eliminate the following drawback, many TCP congestion control algorithms were developed. Most common, oriented for large volumes of data transfers, are the following: HSTCP (RFC 3649), Fast (RFC 3782), STCP (RFC 3286), Cubic [2].

All of these versions also rely on end-to-end ACK and congestion window [3], but general difference is that $W(t)$ is cubic-root on initial phase and depends on latency marginally, comparing with typical versions. The congestion window growth is defined in real-time and depends on last congestion event – received ACK [2]. In other words, the sender is allowed to increase the TCP data rate for each incoming ACK. As a result, unhampered transmission of ACK for successful flow control is very important, and any ACK rate suspension can influence the TCP functionality and performance [4], [5], and [6].

In consequence, the growth of network bandwidth is coherent with ACK message rate on the following TCP operational scheme. When the network capacity is increasing, the ACK rate on the channel is increasing too. So, in high speed network we have another undesirable feature – growth of technological expenditures [7].

Moreover, TCP is mainly used in Internet for bulk traffic (HTTP, FTP, SMTP, IMAP, and etc) [8]. Bulk traffic sources are sending large packets in one direction only [9]. Therefore, uplink traffic is composed of short packets as showed in [4]. In such conditions, encapsulated Ethernet frame with TCP message without data (only ACK flag and number) is 64 Bytes, not including the preamble.

The poor performance of short frame traffic is well known and analyzed in detail [10], and [11]. This degradation occurs for two reasons: technological expenditures [7] and routers CPU overload [12]. The last concerns to the fact that router CPU load depend on the packet rate and not to the packet length [13]. It's clear, that on high speed networks this is observable better [12]. Therefore, with growth of bandwidth the TCP ACK message rate is increasing too. For a large number of short packets in traffic more efficient routers will be needed. Hereby, in following situation the routers with slow CPUs will become bottlenecks and will lead to the degradation of network performance too. The critical threshold of router stability is ~ 45-50% of CPU load [14].

One of the ways to increase performance in modern high speed networks is by lowering the number of TCP ACK messages. This can be done through ACK filtering mechanism [15], which is well known and used in asymmetric and wireless networks environments [4], [5], [6], but is considered to be inconsistent with TCP traffic.

The aim of current paper is to find, how ACK filtering can influence the TCP functionality, and how significant number of TCP ACK messages affects the performance of the network channel – chain of the routers.

TCP acknowledgment mechanism

The acknowledgement mechanism is at the heart of TCP and relies on demand for the receiver to communicate with the sender by sending back an ACK as it receives data. TCP uses a cumulative ACK technique (RFC 813). The ACK number field in a TCP message header received by the sender indicates that all bytes of data with sequence numbers less than that value have been successfully received (ACK'ed) by the destination application (Fig.1. segments *data1*, *data2* and *ack1*, *ack2*).

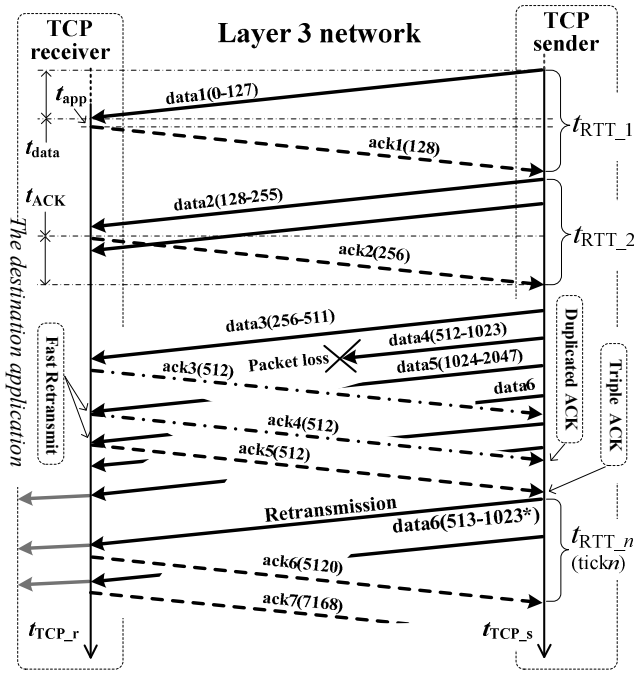


Fig. 1. TCP operation and acknowledgment mechanism diagram

Normally TCP does not send an ACK the instant it receives data. It delays ACK (RFC 2581), hoping to have data going in the same direction. A typical TCP uses delay of 200 ms and sends ACK for every other data message. However, there is not much TCP traffic with bidirectional data flows in Internet [8]. Delayed ACK is usable for an interactive applications (Telnet, SSH, and etc.) only [9].

The TCP sender's uses ACK to estimate how much data can be outstanding in the network without packets being lost in routers queues. For this purpose it uses a congestion window, which growth function $W(t)$ depends on the feedback sender gets from the network through the received ACK. After initializing the TCP sender is allowed to increase the $W(t)$ for each incoming ACK.

In this paper for research we use TCP Cubic where the congestion window is determined by function [2]:

$$W(t)_C = C \left(t - \sqrt[3]{\frac{W(t)_{\max} \cdot \beta}{C}} \right)^3 + W(t)_{\max}, \quad (1)$$

where C is a scaling factor, t [s] – the elapsed time from the last window reduction, $W(t)_{\max}$ [bytes] – the maximal window size before the last reduction; β – a constant multiplication decrease factor applied for $W(t)_C$ reduction to minimal at the time of loss event ($\beta \cdot W(t)_{\max}$) [2]:

$$W(t)_{\min} = W(t)_{\max} - \beta \cdot W(t)_{\max}. \quad (2)$$

Consequently, the ACK reception for TCP sender is very important process. Especially at initial phase, where growth of $W(t)_C$ is very intense [2] and ACK loss can lead to degradation of the congestion window to its minimal value. Meanwhile, on phase when $W(t)_C$ is close or equal to maximal value - $W(t)_{\max}$, growth becomes minimal (2) and congestion event becomes less important. In that case, ACK is mainly used for detection of data message loss and this is a second very important function of the TCP acknowledgment mechanism (RFC 813).

The TCP data message retransmission can occur because of two main reasons [16]: after expiration of retransmission timer (RFC 813) and after receiving three or more duplicated ACK (Fig. 1: *ack3*, *ack4*, *ack5*). In first case the retransmission timer expires when no new data is acknowledged for a set of threshold time (RFC 2581).

The retransmission timeout (t_{RTO}) is taken as a loss indication, and it triggers retransmission of the unacknowledged segments. The threshold time during which the confirming ACK message must be received is:

$$t_{RTO} = \overline{t_{RTT}} + 4 \cdot \sigma_{RTT}, \quad (3)$$

where $\overline{t_{RTT}}$ is the average of time of successful TCP message transmission (Fig. 1: t_{RTT} measurements); σ_{RTT} – the root-mean-square of deviation of t_{RTT} .

If during t_{RTO} the ACK is not received, the data segments loss will be detected and the TCP sender will set $W(t)_C$ to 1 segment (RFC 2988); since t_{RTO} indicate that channel utilization has changed dramatically [16].

Second reason occurs after receiving three or more duplicated ACK [16]. As shown in Figure 1 the message *data4* was lost. The TCP receiver accepts messages *data3*, *data5*, *data6*, and so on (Fig.1.), but not *data4*. The *ack4*, *ack5* with same ACK number 512 were sent for received messages. After duplicated ACK is detected, the transmitter waits for t_{RTO} to expire. TCP does not yet know whether a duplicate ACK is caused by a loss or just reordering of segments. TCP sender waits the timeout (3) and assumes that if there is just a reordering it will not get ACK with number 512 anymore. However, after a while, a third duplicated ACK is received in a row (*ack5*). It is a strong indication that segments have been lost. TCP performs retransmission of segments 513-1023 with *data6*, without waiting for a t_{RTO} to expire. After this the sender maintains the number of outstanding segments by sending a new segment for each incoming ACK (1). It should be noted that only retransmission of identified as lost (timed-out) TCP segments, are implemented in the "conservative" TCP versions (RFC 3517). Meanwhile, in "aggressive" TCP implementation after the loss, all unacknowledged messages are retransmitted [16].

The TCP contains only a general assertion that data should be acknowledged promptly, but gives no more specific indication as to how quickly and as how frequently an ACK must be sent. In RFC's clearly is indicated, that current mechanism must maintain two very important functions: to prevent data retransmission, and as soon as possible to make ACK to permitting further data to be sent. In addition to this argument, the fact that ACK message are very important in the initial phase (1) of data transmission and the fact that rate of ACK rely on data rate (segments loss due to receiving 3 or more duplicated ACK) and cannot be less than $1/t_{RTO}$ (3), must be evaluated.

ACK filtering technique

The concept of ACK filtering was discussed well in [15]. The idea is fairly simple. When router needs to send the current ACK, it scans queue for any early TCP ACK. If ACK already exist in the queue, simply drop them before a queuing the new ACK. Since acknowledgments are cumulative, the newest one obsoletes all older ACK. So,

there's no need for more than one per connection to ever be in the queue. Dropping old ACK when a new one is queued means there would never be more than one ACK on the queue at any time, so this is pretty much the same as replacing the earlier ACK with the newer one.

In the same reference [15] the main drawbacks of ACK filtering were reviewed. It is noted that ACK clocking scheme, which for both drop and congestion control is used, with the ACK filtering can be destroyed. The current propositions are compelling. However, the discussed apprehensions can be challenged with arguments of [4], where the TCP performance in the asymmetric links was analyzed. It is shown that asymmetry affects the TCP performance, because it relies on feedback of cumulative ACK from the receiver. In addition, typical TCP is ACK clocked, so the arrivals of ACK on the reverse channel have significant effects on the forward channel throughput. In the networks with bandwidth asymmetry the ACK filtering can work well. This improves the forward TCP throughput and the fairness of competing connections greatly. In paper ACK filtering was modeled on Opnet.

Work [5] presents a quite similar study. Analysis of unfairness problem between TCP upstream data and ACK downstream on the unevenly shared wireless channel is provided. It is shown that ACK filtering increases 802.11 channel utilization without any dependence on t_{RTO} (3).

The extensive simulations with ACK filtering in [6] were proposed. It is demonstrated that lowering ACK number can improved TCP performance significantly: achieving up to 25% gain in chain networks and 35% in a complex grid network, compare with typical TCP. In work the ACK filtering motivation follows from fact that short ACK messages consume channel capacity comparable to data packets when the transmission is high rate.

The methodology of experiments

In order to find out, what real influence the ACK filtering makes to the TCP functionality and how it affects the performance of the network channel devices (routers) an experiments were performed. For this reason Ethernet network with IP routing, and TCP session between two independent nodes (PC0 and PC1/PC2, Linux OS, 2.6.32 kernel, and TCP Cubic version enabled [2]) were created. The structure of network is presented in Figure 2.

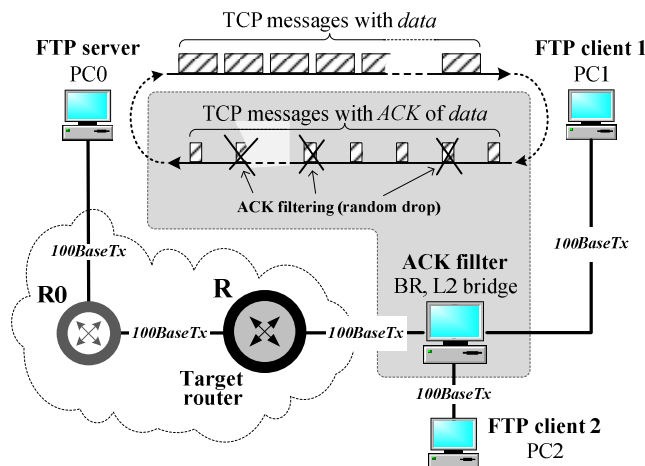


Fig. 2. Structure of network used in experiments

In current network the following equipment has been used: PC0 as FTP server – the transmitter of data TCP messages, and PC1 (PC2) as FTP client – receiver of TCP data messages (TCP ACK sender), the transparent Ethernet bridge – BR, target router – R, and additional router – R0 (Cisco 881) as well. For experiments two routers of different generations Cisco 881 and Cisco 1841 were used. The main difference between them is CPU power. All devices were connected with 100 Mbps Ethernet links (100BaseTx). The ACK filtering was implemented on BR device. It has been created on Linux based (2.6.32 kernel) PC with Ethernet bridging `<brctl>` application. This tool was taken because Linux bridging is faster, work as simple switch, and don't make significant impact to flow parameters comparing with routing.

The ACK filtering has been pursued only in one direction from PC1 to PC0, whereas in opposite direction the traffic passed through BR without any alterations. ACK filtering was based on exact frame rate control with Committed Information Rate (CIR) and Committed Burst Size (CBS=5 kB). The last one was used to avoid degradation of congestion window on initial growth phase (1). Filtering was made using `<tc>` application, which drop/policed ACK messages if it exceeds specified rate. The scripting code of ACK filtering is shown in Figure 3.

```
01 tc qdisc add dev eth1 ingress
02 tc filter add dev eth1 parent ffff:0
   protocol all prio 1 u32 match u32
   0xaff0001 0xffffffff at 16 classid
   ffff:0 police index 2 rate 12500bps
   burst 102400 mpu 0 action drop/pass
03 tc filter add dev eth1 parent ffff:0
   protocol all prio 1 u32 match u32
   0x0 0x0 at 0 classid ffff:0 police
   index 3 rate 1bps burst 1 action
   drop/drop
```

Fig. 3. Scripting source of tc policing

The second router (R0) has been used on purpose to keep more realistic IP based network with all routing and switching functionalities. The target router parameters during the experiments with SNMP protocol were collected throw independent router interface. For data transmissions the FTP application has been used. In all experimental iterations file of 400 MB size was transferred. The data speed was controlled on PC0 (FTP server) with `<tc>` script, which shaped to desirable speed without packet loss (delay of traffic only). For this Token Bucket Filtering – TBF (RFC 5624) was used.

Performance evaluation experiments

The target of first experiment scenario was to find how the ACK filtering can influence the TCP performance and data transfer integrity. For this, a file of 400 MB size from PC0 to PC1 was repeatedly transferred (Fig. 2.). On the each iteration the ACK filtering rate (0%, 20%, 40%, 60%, and 80%; values of the maximal ACK rate without filtering) was changed.

As shown in Figure 4 TCP message rate for entire period was stable in all iterations. The growth of message rate (1) is high and equal at all ACK drop values (events up to ~3s). This occurred because of two reasons: $W(t)_C$ of

used TCP version slightly depends t_{RTO} (3) and ACK filtering is activated only after CBS is exceeded – when the $W(t)_C = W(t)_{\max}$ (1). At the end of transfer we have “decline” – the finish of data transfer.

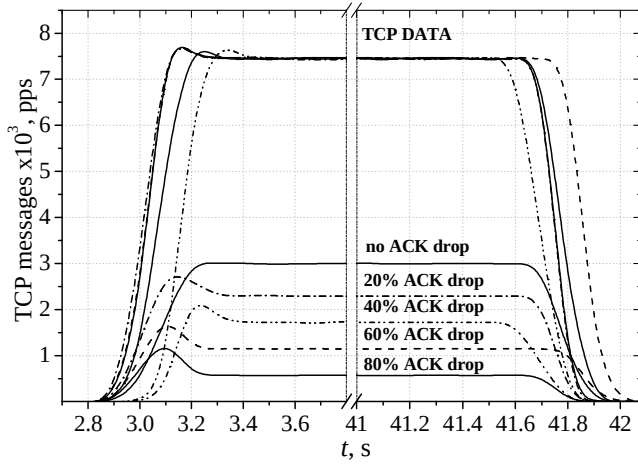


Fig. 4. TCP messages rate during file transmission for various ACK drop values

The result shows that ACK filtering does not affect data transfers of single TCP session. We observed that transfer remains stable for up to 80% of ACK drops. However, if losses are above 85% the ACK rate becomes less than $1/t_{RTO}$ (3), and data rate degrades fatally.

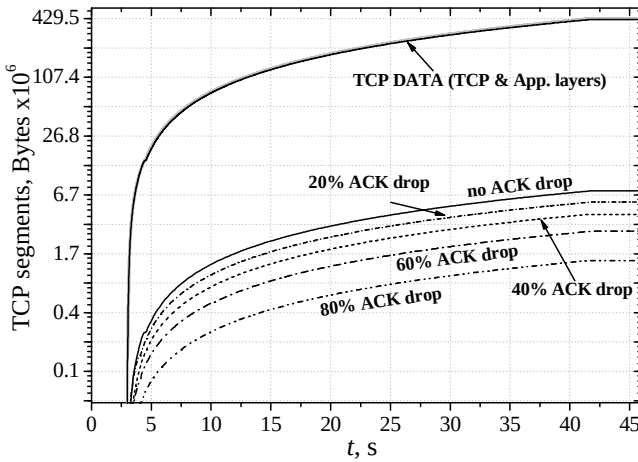


Fig. 5. Cumulative growth of TCP segments count during file transmission for various ACK drop values

The comparison of cumulative TCP segments (count of segments on TCP layer) and FTP bytes (count of data on FTP layer) rates during period (t) of file transfer is presented in Figure 5. It is shown that on FTP and TCP layers (lines are coincident) the same amount of bytes was received at any given time period. Consequently, it is possible to do the suggestion, that count of duplicated ACK and TCP data retransmissions (3) are not increased respectively (Fig. 1: $t_{RTT} = t_{data} + t_{ACK}$ is less then t_{RTO}).

The second experiment goal was to find what kind of influence cumulative ACK mechanism has on network equipment performance. For this purpose a file transfer of 400 MB, was performed and CPU load of router (Cisco 881, Cisco 1841) was measured (Fig. 2.). The experiments were performed at various data rates (1, 4, 8, 16, 32, 65,

and 90 Mbps) in scenarios with and without filtering and in scenario when 80% of ACK is filtered.

As shown in Figure 6 the CPU load is decreasing when the ACK filtering is used. The same situation is observable for both routers. It is clear that results do not depend on router type and amount of data in TCP message. It depends on amount of processed packets by the router CPU. This is confirming the results presented in [13]. Moreover, in Fig. 6 is observable that CPU load utilization is linear, and depends on frame rate: if the TCP data rate increases, the ACK rate is increasing too. The functions of the CPU load curves are quite similar for both routers. The difference is only in designed CPU's power.

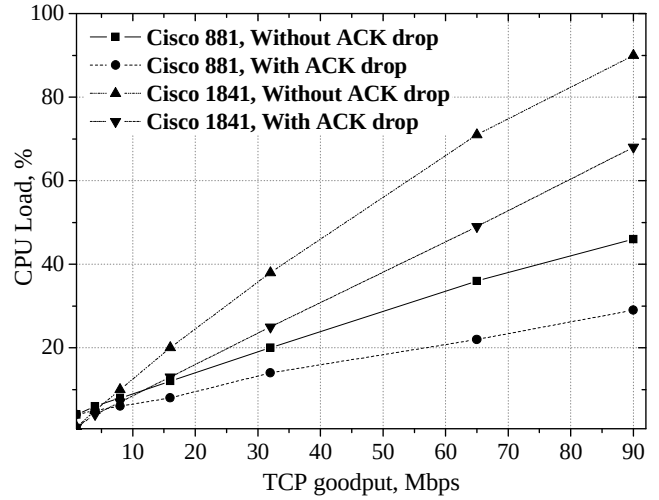


Fig. 6. Target routers CPU load for various FTP traffic rate (TCP goodput) when 80% ACK is dropped

The relation between target router performance and CPU load is shown in Figure 7. Performance increase should be understood as relative CPU load reduction caused by employing of ACK filtering. With ACK drop of 80% the performance can be increased by 30%, comparing with case when CPU load is 25%, and ACK filtering is not used. Meanwhile, if we have the CPU load of 60% (more than routers overload threshold [13]) with 80% of ACK drop we can increase the performance by 32%. It means that in current situation CPU load will be approx. 40%.

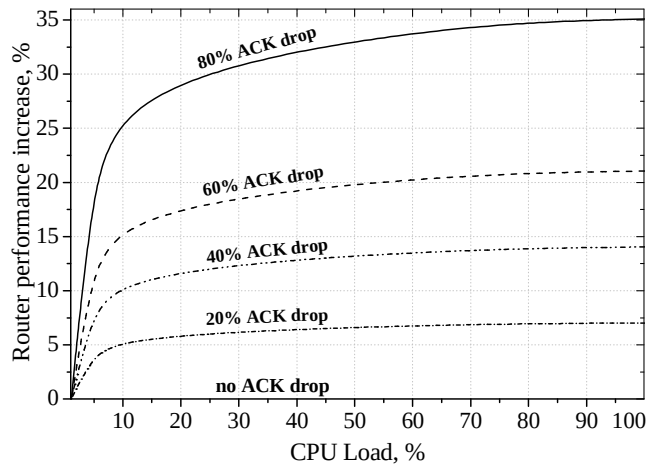


Fig. 7. Performance increase on target router CPU load for various ACK drop values

The third experiment issue was to find how the ACK filtering can influence two concurrent and independent TCP sessions. During investigation the PC0 was used as FTP server and PC1/PC2 - as FTP clients. With each TCP session the 400 MB files were transferred. The ACK filtering was performed on both sessions simultaneously.

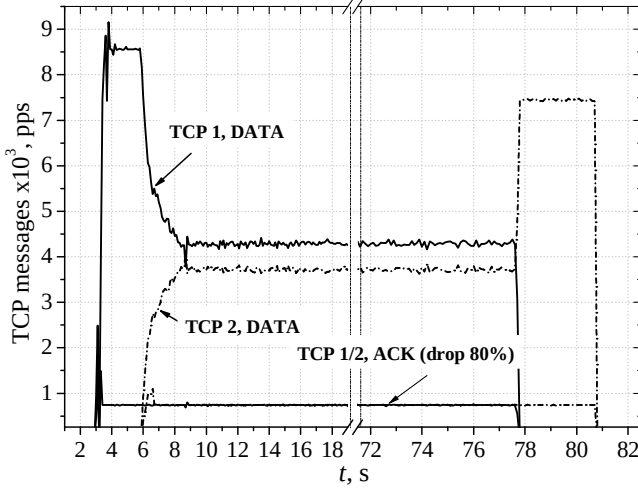


Fig. 8. TCP message rate during two files transmission: both sessions with 80% of ACK drops.

The dependence of two TCP sessions message rate during file transmissions is presented in Figure 8. Graph clearly shows that both independent sessions remain concurrent and divide channel almost equally. In fact, this cannot be so, because TCP objective is channel utilization, and not equal channel sharing (QoS function). At 3s, as in situation with one session (Fig. 4), the TCP1 begins to increase the message rate according to $W(t)_C$ (1). While at 6s TCP2 session is starting too, and about 9s the message rate of both sessions becomes approx.: $R_{TCP1} \approx 44\text{Mbps}$; $R_{TCP2} \approx 49\text{Mbps}$; $R_{TCP} \approx 93\text{Mbps}$. While the TCP1 session is degraded due to congestion (2).

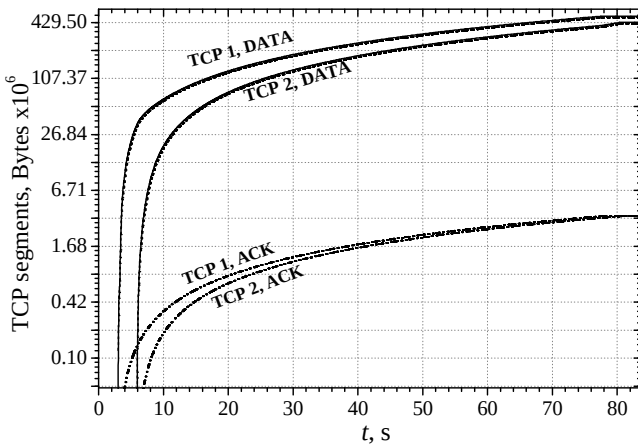


Fig. 9. Cumulative growth of TCP segments count during two file transmission: both sessions with 80% of ACK drops

After 78s the TCP1 rate is decreasing since finishing of file transfer, while the TCP2 conversely starts to grow-up rate (1). At 81s TCP2 is finishing transfer too. Current fine competition between two independent session's show, that both of them from TCP point of view are working well, and the ACK filtering does not make significant influence on TCP stability in current conditions.

The dependence of cumulative data during two file transmissions is shown in Figure 9. It can be seen that both sessions collect messages well in TCP layer and in application layer (Fig.1.). It means that TCP goodput (transmitted data to upper layer) of both sessions are close to maximal, while the count of duplicated ACK is minimal.

High interoperability of TCP defines whether a protocol is fair to other TCP versions. Therefore, it's important to find how unfairly the ACK filtering affects most common TCP. For this purpose the fourth experiment was performed. The scenario was the same as in previous, only the ACK filtering for one session and the PC0 without shaping were used (Fig. 2.).

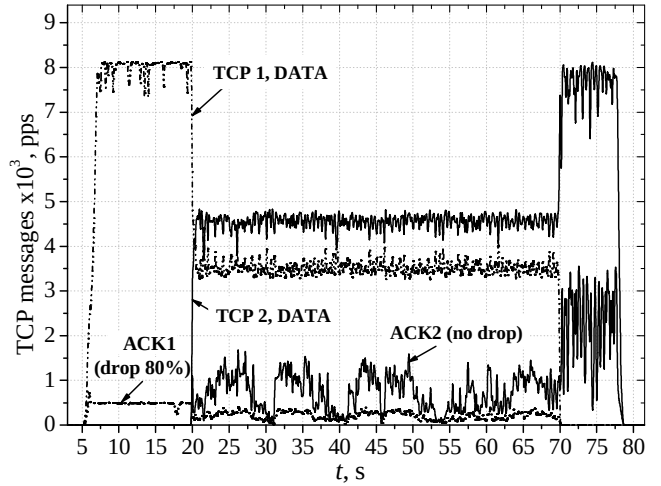


Fig. 10. TCP message rate during two files transmission: one session without ACK drop, and other with 80% of drops

The dependence of message rate of two TCP sessions on file transmissions time is presented in Figure 10. Graph shows that both sessions divide channel almost equally as in previous scenario. The insignificant unfairness between sessions was observed. But this is typical case for real network. To understand unfairness better, the extensive analysis must be done with various TCP versions. And this is the main issue of future investigations.

Conclusions

In current work the investigation of TCP functions and ACK filtering was presented. It is shown that TCP congestion window and acknowledgment mechanism are dependent processes with common variables. Therefore, the growth of network bandwidth is coherent with ACK rate: when the network capacity is increasing, the ACK rate on channel is increasing too. This leads to network equipment CPU performance degradation. To resolve this weakness the usage of ACK filtering was proposed.

The experiments were performed to find how significantly number of ACK affects the router CPU and how impact of ACK filtering can influence the TCP functionality and performance. We conclude that:

1. ACK filtering does not significantly affect the TCP data transfer in normal network conditions. The TCP data transfer remains stable for up to 80% of ACK drops. However, if losses are above 85% the ACK rate becomes less than $1/t_{RTO}$, and session is terminating immediately.

2. The performance of router CPU depends on ACK count and can be increased with ACK filtering. The CPU load utilization is linear, and depends on data rate: if the TCP data rate increases, the ACK rate is increasing too.

3. On CPU load of 25% with 80% of ACK drop, it is possible to increase the router performance by 30%. Meanwhile, on CPU load of 60%, on the same filtering conditions, the performance can be increased by 32%.

4. The ACK filtering can be used not only with single TCP session but also with concurrent sessions. Results show that two sessions work well, without any evident signs of the instabilities; although a slight unfairness among TCP sessions were observed.

In conclusion, the presented results show that ACK filtering does not affect the stability of TCP in normal network conditions, but allows more efficient use of system and network resources. Therefore, for real implementation of suggested solution the influence of the ACK filtering on interoperability with other TCP versions, unfairness problem, and etc. the extensive analysis with other popular TCP versions must be done. These issues are the main of the future investigations.

References

- [1] Dalton L. A., Isen C. A Study on High Speed TCP Protocols // Proc. IEEE: GLOBECOM 2004, – Vol. 2. – P. 851-855.
- [2] Sangtae H., Injong R., Lisong X. CUBIC: A New TCP-Friendly High-Speed TCP Variant // Proc. IEEE: SIGOPS, 2008. – Vol. 42, No. 5. – P. 64-74.
- [3] Jacobson V. Congestion avoidance and control // in Proc. Conference IEEE: SIGCOMM, Vol. 1, 1988. – P. 314-329.
- [4] Wu' H., Wu' J., Cheng' Sh., Ma J. ACK Filtering on Bandwidth Asymmetry Networks // Proc. IEEE: APCC/OECC, 1999. – P. 175-178.
- [5] Keceli F., Inan I., Ayanoglu E. TCP ACK Congestion Control and Filtering for Fairness Provision in the Uplink of IEEE 802.11 Infrastructure Basic Service Set // in Proc. Conference IEEE: ICC, 2007. – P. 4512-4517.
- [6] Chen B., Marsic I., Shao H-R., Miller R. Improved Delayed ACK for TCP over Multi-hop Wireless Networks // Proc. IEEE: WCNC, 2009. – P. 1-5.
- [7] Kajackas A., Pavilanskas L. Analysis of the Technological Expenditures of Common WLAN Models // Electronics and Electrical Engineering. Kaunas: Technologija, 2006. – No. 8(72). – P. 19-24.
- [8] Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K. Internet traffic classification demystified: myths, caveats, and the best practices // In Proc. IEEE: CoNEXT, New York: ACM, 2008. – P. 1-12.
- [9] Caceres R., Danzig P. B., Jamin S., Mitzel D. J. Characteristics Of Wide-Area TCP/IP Conversations // Proc. IEEE: ACM SIGCOMM, 1991. – P. 101-112.
- [10] Xiao Y., Rosdahl J. Throughput and delay limits of IEEE 802.11 // IEEE: Comm. Letters, Vol.6, 2002. P. 355-357.
- [11] Chatzimisios P., Vitsas V., Boucouvalas A. C. Throughput and delay analysis of IEEE 802.11 protocol // Proc. IEEE: Networked Appliances, 2002. – P. 168- 174.
- [12] Bencivenni M., Carbone A., Fella A., Galli D., Marconi U., Peco G., Perazzini S., Vagnoni V., Zani S. High rate packet transmission on 10 Gbit/s Ethernet LAN using commodity hardware // Proc. NPSS: Real Time, , 2009. – P. 167-182.
- [13] Paredes-Farrera M., Fleury M., Ghanbari M. Router Response to Traffic at a Bottleneck Link // in Proc. 2nd Int. Conference IEEE: TRIDENTCOM, 2006. – P. 4-41.
- [14] Ogawa Y., Nakaya A. Estimating the Performance of a Large Enterprise Network for the Updating of Routing Information // in Proc. Conference Workshop IEEE: IP Operations and Management, 2002. P. 161-165.
- [15] Karn P. at all. Mailing list, Subject: dropping TCP ACK's // end2end-interest, karn@qualcomm.com., 1996, Internet: /ftp://ftp.isi.edu/end2end/end2end-interest-1996.mail [Accessed on: 2009-11-30].
- [16] Jacobson V. Congestion avoidance and control // in Proc. Conference SIGCOMM' 88: ACM Special Interest Grp. on Data Comm., New York: ACM, 1988. – P. 314-329.

Received 2010 02 07

L. Pavilanskas, A. Statkus. Evaluation of TCP acknowledgment mechanism influence on router performance // Electronics and Electrical Engineering. – Kaunas: Technologija, 20XX. – No. X(XX). – P. XX-XX.

With growth of Internet traffic, the efficient usage of congestion control in heterogeneous networks is essential to successful utilizing resources. The main protocol for that purposes is TCP which relies on end-to-end acknowledgments and maintains a congestion window. The ACK mechanism is at the heart of TCP. In this paper the detailed analysis of ACK mechanism functions and possibilities to reduce the ACK message count with ACK filtering were performed. The presented results of experiments with real network are shows that router performance, without any considerable influence to TCP functionality can be significantly increased when the ACK filtering is used. Ill. 10, bibl. 16 (in English, abstracts in English, Russian and Lithuanian).

Л. Павиланскас, А. Статкус. Оценка влияния механизма TCP подтверждения на производительность маршрутизатора // Электроника и электротехника. – Каунас: Технология, 20XX. – № X(XX). – С. XX-XX.

При увеличении Интернет трафика эффективное управление перегрузками в гетерогенных сетях является основной приметой успешного использования ресурсов. TCP является наиболее подходящим протоколом для этих целей, который, при помощи механизма подтверждения и алгоритма окна переполнения реализует надёжную передачу данных. Механизм подтверждения является основным в TCP. В этой статье детально анализируются функции этого механизма и возможности уменьшения сообщений ACK при помощи фильтрования. Предоставленные эксперименты на реальной сети подтверждают, что используя фильтрование ACK возможно значительно увеличить производительность маршрутизаторов без воздействия на стабильность TCP. Ил. 10, библи. 16 (на английском языке; рефераты на английском, русском и литовском яз.).

L. Pavilanskas, A. Statkus. TCP patvirtinimo mechanizmo įtakos maršrutizatoriaus našumui įvertinimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 20XX. – Nr. X(XX). – P. XX-XX.

Didėjant Interneto srautams, efektyvus kanalų perkrovų valdymas heterogeniniuose tinkluose yra svarbiausias veiksnys sėkmingai išnaudojant duomenų perdavimo tinklų įrenginius. Tinkamiausias ir populiariausias šiai funkcijai atlikti yra TCP protokolas, kuris patvirtinimo mechanizmo ir persipildymo lango algoritmo pagalba realizuoja patikimą duomenų perdavimą. Patvirtinimo mechanizmas yra TCP funkcionavimo pagrindas. Šiame straipsnyje detalai analizuojamos šio mechanizmo funkcijos, bei galimybės, panaudojant filtravimo sprendimus, sumažinti generuojamų ACK pranešimų skaičių. Pateikti realiaje tinkle atliktų eksperimentų rezultatai rodo, kad panaudojus ACK filtravimą, bei nedarant ženklios neigiamos įtakos TCP stabilumui, galima žymiai padidinti tinklo maršrutizatorių našumą. Il. 10, bibl. 16 (anglų kalba; santraukos anglų, rusų ir lietuvių

